Radio-Wave Propagation Prediction using Ray-Tracing Techniques on a Network of Workstations (NOW)[*]

Zhongqiang Chen
Department of Computer
& Information Science
Polytechnic University
Brooklyn, NY 11201
zchen@milos.poly.edu

Alex Delis
Department of Informatics
& Telecommunications
University of Athens
Athens, 15771, Greece
ad@di.uoa.gr

Henry L. Bertoni
Department of Electrical
& Computer Engineering
Polytechnic University
Brooklyn, NY 11201
hbertoni@duke.poly.edu

October 2004

Abstract

Ray-tracing based radio wave propagation prediction models play an important role in the design of contemporary wireless networks as they may now take into account diverse physical phenomena including reflections, diffractions, and diffuse scattering. However, such models are computationally expensive even for moderately complex geographic environments. In this paper, we propose a computational framework that functions on a network of workstations (NOW) and helps speed up the lengthy prediction process. In ray-tracing based radio propagation prediction models, orders of diffractions are usually processed in a stage-by-stage fashion. In addition, various source points (transmitters, diffraction corners, or diffuse scattering points) and different ray-paths require different processing times. To address these widely varying needs, we propose a combination of the phase-parallel and manager/workers paradigms as the underpinning framework. The phase-parallel component is used to coordinate different computation stages, while the manager/workers paradigm is used to balance workloads among nodes within each stage. The original computation is partitioned into multiple small tasks based on either raypath-level or source-point-level granularity. Dynamic load-balancing scheduling schemes are employed to allocate the resulting tasks to the workers.

We also address issues regarding main memory consumption, intermediate data assembly, and final prediction generation. We implement our proposed computational model on a NOW configuration by using the message passing interface (MPI) standard. Our experiments with real and synthetic building and terrain databases show that, when no constraint is imposed on the main memory consumption, the proposed prediction model performs very well and achieves nearly linear speedups under various workload. When main memory consumption is a concern, our model still delivers very promising performance rates provided that the complexity of the involved computation is high, so that the extra computation and communication overhead introduced by the proposed model do not dominate the original computation. The accuracy of prediction results and the achievable speedup rates can be significantly improved when 3D building and terrain databases are used and/or diffuse scattering effect is taken into account.

Indexing Terms: Ray-tracing based radio propagation prediction model, Radio wave prediction model on Network of Workstations (NOW), Data or control domain decomposition, Dynamic workload-balancing scheduling schemes.

[*]This work was partially supported by NSF under grant IRI-9733642, the U.S. Department of Commerce under grant 4000186, and The New York Center for Advanced Technology in Telecommunications (CATT).

1 Introduction

Ray-tracing based radio wave propagation prediction models play an prevalent role in the design of modern wireless networks [32, 50, 77, 46, 58]. The main objective of such models is to efficiently yield accurate predictions on radio wave propagation pertinent statistics including received signal strengths for mobile locations, delay spread, and angle of arrival. At the same time, it has been recognized that these models are computationally very expensive and require a considerable amount of processing time to attain reasonable accurate prediction results [7, 53]. In this context, it is typical that ray-tracing based models take hours to generate predictions for moderately sized geographic areas such as 1 km^2 [53, 50]. The core pincushion method (or otherwise known as the shoot-and-bounce method) is the main source for computational intensity for the models in discussion [46, 53]. In this method, rays are launched with an angular separation δ from source points, which are either transmitters or diffraction corners acting as secondary sources. Each raypath may encounter reflections, diffractions, refractions (transmissions), and diffuse scattering. To achieve reasonable prediction accuracy, the angular separation δ needs to be very small and usually less than 0.6° (or about 0.01 radians [7]). Consequently, the number of raypaths between base stations and mobile stations may be explosive and extremely long CPU processing times are required to examine all the raypaths in question. As the coverage of a wireless system increases, and the corresponding network environment becomes more complex, the interactions between raypaths and geometric objects including various types of buildings, terrain, and vegetation make matters worse calling for even more dramatic increases in computation [32, 7, 50].

A number of approaches have been proposed to shorten the computation time for prediction models. The complexity of building databases can be reduced by simplifying footprints [16]. Data filtering and cleansing techniques have been proposed in [50]. Procedure-approximation methods are also employed to address the same problem [15]. In these methods, either a subset of raypaths are processed based on different requirements for prediction accuracy [15] or different physical phenomena such as vertical plane diffractions, multi-paths, and existence of vegetation are exploited to offer balance between prediction accuracy and computation time [50]. Both data-reduction methods and procedure-approximation methods have a common drawback: they trade prediction accuracy for processing time.

A natural way to overcome the above drawback is to use the parallel and distributed computation techniques to speed up computations, while keeping the accuracy intact [40, 41, 17, 3]. More specifically, the usage of a network of workstations (NOW) is particularly attractive as such computer system configurations are readily available at this time. By resorting to such parallel/distributed computation methods, our main objective is to both distribute and/or parallelize various components of our ray-tracing prediction model among multiple nodes in such a way that the processing time will decrease proportionally to the number of nodes involved. Should the latter be feasible, we anticipate significant gains in the radio wave propagation prediction area. There are many advantages to use NOW-based computation techniques in a ray-tracing based radio propagation prediction model. The accuracy of prediction results is not affected since neither data reduction nor procedure approximation methods are introduced. The relationship between computation resources (i.e., the number of nodes involved) and prediction time is straightforward as more resources yield shorter response times. The NOW-based computation techniques also improve overall scalability of the model used. In the data reduction methods, the prediction time is not linear to the data complexity at hand while in the procedure-approximation methods, the requisite time mostly depends on the accurate estimate of each raypath's contribution to the prediction results. Finally, NOW-based ray-tracing models are much more flexible since new nodes can dynamically join existing sites to help the ongoing computation. On the other hand, the design and analysis of a NOW-based prediction model involves the examination of a complex set of interrelated issues such as computational concurrency, computing unit (task) granularity, task allocation and scheduling, communication and synchronization, as well as workload-balancing. It is this diversity of frequently competing factors that make the design and implementation of NOW-based algorithms for radio propagation prediction a challenging task. Clearly, there are tradeoffs among the above interrelated factors. Metrics used to measure and evaluate the performance of a NOW-based system attempt to capture and quantify most of such factors and include the nature of workloads processed, such as speedup, workload expansion ratio, and resource utilization.

In this paper, we investigate these interrelated factors, and establish a feasible and effective parallel/distributed computation model for radio wave propagation prediction. We use a combination of phase parallel and manager/workers paradigms to coordinate, synchronize, and distribute computation, and employ dynamic load-balancing scheduling scheme to allocate workload among nodes. The metric of speedup is the main performance index as our overall design goal is to generate predictions as quickly as possible. In addition, we monitor other performance indicators including workload expansion ratio, efficiency, isoefficiency, and resource utilization. Our experiments show that the proposed NOW-based radio propagation prediction model presents consistent performance rates across a wide range of workloads, and achieves almost-linear speedup rates in most examined cases.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 discusses the baseline ray-tracing based radio propagation prediction algorithm and examines the unique characteristics of the problem at hand. Section 4 outlines our proposed NOW-based parallel/distributed computational model, computation decomposition methods, and workload-balancing schemes. Techniques to further improve the efficiency of the proposed model are discussed in Section 5. Section 6 presents a comprehensive experimental evaluation of our model while concluding remarks and future work can be found in Section 7.

2 Related Work

In order to address computationally intensive problems, NOWs have been used in a wide range of fields including large-scale databases, scientific computations, computer graphics, multimedia, wave propagation predictions, and telecommunication systems simulations [18, 62, 54, 56, 2, 39, 71, 59].

The fundamental idea in NOW database systems is to carry out simultaneous I/O operations whenever possible and to execute highly intensive CPU processing in a distributed fashion [43, 66]. Computationally intensive problems such as the N-body problem have been addressed in the context of NOW environments. Here, the movement of a set of particles is simulated under the influence of gravitational, electrostatic and Vander Waals attractions [69, 10, 36]. Two prevalent forms of the N-body problem known as the Barnes-Hut and the Fast multi-pole (FMM) methods have been implemented using message passing and shared-memory architectures [11, 31, 54, 74, 47, 57, 37]. In this context, computation-partitioning and workload-balancing scheduling approaches have been proposed in [69].

One field closely related to our work is the ray-tracing illumination models used in computer graphics that are known to generate high-quality images but suffer from long rendering times [72, 60, 65]. A ray-tracing illumination model launches a very large number of raypaths in a scene consisting of many geometric objects. Each raypath is tested for intersection with objects to determine the visibility of their surfaces, meanwhile, pixel intensities are generated for the produced image. Due to the inherent computation costs, such illumination models have been traditionally used off-line. However, the appearance of networks of processors or workstations has provided the needed computational framework for on-line and interactive ray-tracing illumination models. For instance, an airplane and missile simulator uses a 96-processor SGI Power-Challenge cluster to create real-time effects [55, 56]. Similarly, the Utah ray-tracing system offers interactive capability by utilizing a multi-processor supercomputer with static workload-balancing scheduling scheme to minimize synchronization overhead [60]. The parallel ray-tracer proposed in [27, 26] uses a NOW as its computation platform. Its main assumption is that every participating workstation can retain the entire scene in its main memory at all time. The parallel radiance model in [67] also uses message passing method to communicate among processors and assumes that the entire scene of operation resides in every node's main memory throughout the computation. Additional "parallel" and interactive ray-tracing systems are discussed in [65, 45, 63, 64]. All these systems consider their computational problem as an (nearly) embarrassingly parallel one that can be divided into a number of completely independent and equally intensive components and each such component can be designated to any processor (or machine). Therefore, the communication and/or synchronization overhead is considered minimal [76, 25].

The ray-tracing based radio wave propagation prediction model discussed in this paper differs significantly from conventional ray-tracing illumination models in a number of aspects. First, the objective of illumination models is to create photo-realistic images that focus on visual effects, such as texture and color while radio wave propagation prediction systems put great emphasis on the numerical evaluation of ray qualities, such as field amplitude and time delay. Illumination models are mostly based on diffuse reflections and occasionally specular reflections with only a limited number of such events along the raypath. On the other hand, radio propagation prediction models use reflections, diffractions, and diffuse scattering, and often account for multiple events along a raypath. The inclusion of diffraction forces prediction models to function in "stages" (or phases) of computations. There is also a strong correlation among different raypaths in radio propagation prediction models. The energy carried by a raypath can be known only after all diffraction corners along the raypath are found and their strengths are determined. Consequently, the processing of various raypaths is not independent as is the case in traditional illumination models. More importantly, the processing time for different raypaths can be quite different and the variance is large in radio prediction models compared with their illumination counterparts. The above characteristics lead to the observation that the ray-tracing based radio propagation prediction problem discussed in this paper is not an embarrassingly parallel/distributed problem (not even a nearly embarrassingly parallel/distributed one) as is the case with illumination models [60, 55, 56]. Thus, illumination model techniques cannot be directly applied to radio wave propagation prediction models.

Modeling three-dimensional sound wave propagation is critical for applications such as concert hall design [6, 5, 51], virtual reality [20, 13], and interactive systems [29, 24]. Ray tracing techniques are widely used to find the sound wave propagation paths representing different sequences of reflections, transmissions, and diffractions at surfaces of the environment. The effect of such propagations is the reverberation at the receiver [52, 48]. In contrast to illumination models, more orders of reflections are computed to account for the large range of audible sound frequencies requiring heavy computations [28]. When the wavelength of the sound wave is similar to the geometric feature size, diffraction becomes an essential effect [7, 61]. The wavelengths of audible sound range between 0.02 and 17 meters (for 20 KHz and 20 Hz, respectively, with sound speed of 343 meters per second). Therefore, diffraction effects are considered only for low frequency sound waves and environments where there are large obstacles between source and listener [70, 44]. The diffraction effects and late reverberations are routinely modeled with statistical, perceptual approximations [68, 1]. Even in ray-tracing based acoustic models, the diffraction effect is applied only to a small portion of the environments described with little geometric detail. Thus, most techniques used in traditional illumination models can be applied directly to sound wave models [28]. In contrast, radio wave propagation models examined in this paper base their computation on all pertinent physical phenomena such as specular reflections, diffuse reflections, and diffractions [7].

Recently, there have been efforts to parallelize/distribute CPU intensive simulation tasks for telecommunication systems. An outdoor propagation model for microcells is parallelized by using a Cray T3E supercomputer in [39, 38]. Message passing and the workpool paradigm are used to communicate and balance workload among nodes. However, experimentation shows that the achieved speedup is far from linear. In [71, 34], a parallel ray tracing system is used to optimize the placement of transmitters in an indoor wireless system. A 200-node Beowulf NOW with each processor locally having a complete copy of the building database is employed to carry out the optimization considering only reflections and transmissions (penetrations). In [59], the *FastScat* system is proposed to parallelize electromagnetic scattering calculations on a SGI Origin-2000. The system is implemented in a threaded style assuming a cache-coherent distributed shared memory. Selected data structures are replicated on every processor and data locality is exploited to achieve scalability. Experiments show that speedup of 25.9 is attained for 32 processors.

3 Observations on Sequential Ray-Tracing Prediction Models

In modern wireless network design, ray-tracing models are used to predict the received powers at various locations in the coverage area of a base station so that quality of service is guaranteed. To attain good prediction accuracy, reflection, diffraction, and diffuse scattering should be considered. Suppose that a cellular network is established in Rosslyn, VA as depicted in Figure 1. The base station is located atop a building at the center of the area (location Tx5) and three receiver locations are marked as spots 1, 2, and 3 respectively. Each of these spots may be “reached” with the help of the three propagation planes shown in Figure 1. However, none of these locations can be reached directly as no line-of-sight (LOS) can be established between the base station and the receivers [7]. A raypath that is part of Plane 2 is essentially of non-line-of-sight nature consisting of one reflection and one diffraction at an horizontal building edge to reach spot 2. As no line-of-sight exists between the base station and spots 1 and 3, horizontal or vertical edge diffraction becomes the dominant contributor to the received power at the spots in question. More specifically, a raypath that is part of Plane 3 encounters two vertical diffractions in order to finally reach spot 3. Surfaces illuminated by the base station scatter rays in all directions as well. These scattered rays may reach receivers and contribute to the received powers.

Algorithm 1 can be used to compile radio wave propagation prediction results in a geographic area (as the one depicted in Figures 1 and 2). This baseline ray-tracing based prediction algorithm consists of five distinct stages. In the initialization phase, key user parameters about the prediction settings and operating environment are obtained. The terrain/building databases are initialized and locations of transmitters and receivers, antenna patterns for base stations and mobile stations, carrier frequency, as well as maximum

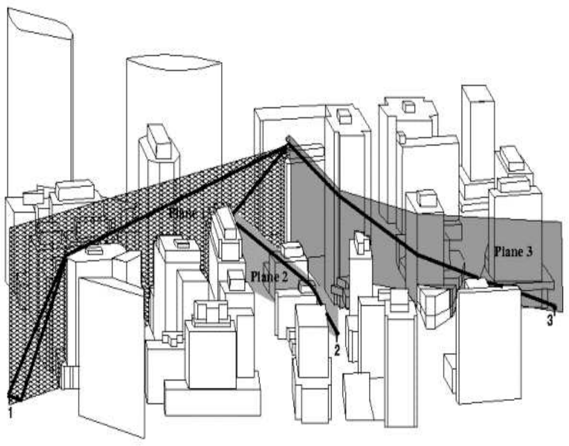


Figure 1: Prospective view of buildings in Rosslyn, VA depicting three radio propagation planes created by launching from a base station (at location Tx5)

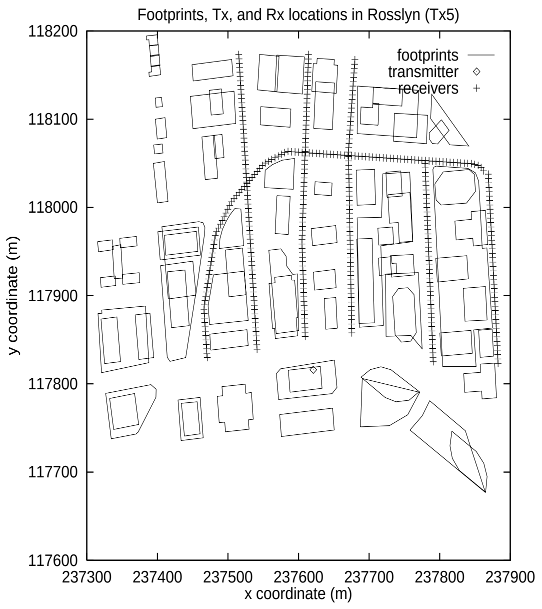


Figure 2: Building footprints, 400 receiver locations, and a transmitter (Tx5–appearing atop of a back-row building in Figure 1) in Rosslyn, VA

numbers of reflections and diffractions are provided (see Figure 2). The next stage deals with transmitters. As raypaths from all transmitters are traced, all illuminated receivers and first order diffraction corners are determined. In the subsequent stage, orders of diffraction corners are processed. All diffraction corners found act as secondary sources and are treated as if they were transmitters themselves. All illuminated receivers are determined. This step is repeated for each order of diffraction; in practice, two orders of diffractions are adequate. After that, all diffuse scattering points are determined and processed as if they were transmitters. The final stage is to generate prediction results. Intermediate results generated by all previous phases are assembled to form predictions, such as received powers by all receivers, delay and angle spreads. Prediction accuracies are found to be adequate for communication system planning–see for example the comparison of predictions with measurements for low base station antennas in Figures 15, 23–26.

It is rather evident from its description that Algorithm 1 has its own intrinsic characteristics that cannot be easily distributed and/or parallelized. Both initialization and prediction results generation stages are rather sequential in nature as they involve I/O operations. The other three stages have to be executed in a synchronized fashion. For example, the output of the transmitters processing is the input of the diffraction corners processing and Bernstein's conditions are formed [41]. There is also control dependency between different stages. Although it is possible to use pipeline techniques to overlap the executions of different stages, the complexity of the resulting system will increase dramatically.

Should the function of Algorithm 1 be distributed, a number of hidden costs should be taken into considerations as well. Firstly, additional computational control has to be introduced. Broadcasting of building

Algorithm 1 Sequential Radio Propagation Prediction Algorithm

```
1: read building database into main memory, cluster all buildings into grids; read configuration file or get
   input about settings from the user interactively
2: for (each transmitter) do
3:     trace its raypaths sequentially; determine illuminated receivers and diffraction corners
4: end for
5: for (each diffraction level $l$) do
6:     for (each secondary transmitter $i$ at level $l$) do
7:         trace raypaths emitted by $i$; determine illuminated receivers and diffraction corners
8:     end for
9: end for
10: generate all the diffuse scattering points
11: for (each diffuse scattering point) do
12:     treat it as a transmitter and trace its raypaths upto maximum number of reflections and diffractions.
13: end for
14: output prediction results
```

database and configuration parameters, collection of intermediate results from all nodes to a single node or vice versa are just some examples. The establishment of lookup tables on various computing nodes that are deemed indispensable in the compilation of radio propagation prediction is another example. Secondly, some communication overhead is unavoidable when message passing is used. Since the problem we study is not embarrassingly parallelizable, many inter-node communications have to take place in order to coordinate, synchronize, and exchange information among different sites. Finally, main memory consumption is of vital importance in a NOW-based solution as the full duplication of data in every site is not necessarily deemed a viable option at all times. In the course of tracing raypaths, partial data replication is required since any processing unit ultimately needs information about the same buildings and environmental obstacles involved in a computation. Because of the hidden overheads, it is challenging to achieve speedup rates that are linearly proportional to the size of NOW.

Without taking any overheads into account, the best achievable speedup for a computation is governed by the fraction α of the workload that must be executed sequentially as Amdahl's law points out [4]. Suppose the portion $(1-\alpha)W$ of a workload W can be perfectly parallelized, then the attained speedup is $S_n = n/[1+(n-1)\alpha]$. In the case of a very large NOW with $n \to \infty$, Amdahl's law yields the best possible speedup rate of S_n=$1/\alpha$. The value of α is determined by the intrinsic features of the specific computation and in our case mainly by the angular separation δ between neighboring rays. In the presence of tasks with light CPU requirements, Gustafson's law [33] offers a flexible mechanism for better use of computing resources. By pursuing a "higher resolution" setting of the problem at hand (i.e., by reducing the angular separation in our case), one can expand the workload and improve the quality of results obtained. Overall, the ray-tracing based radio propagation prediction problem is parallelizable for the transmitters, diffraction corners, and diffuse scattering points processing stages. However, it is not embarrassingly parallelizable due to its intrinsic serial parts and data/control dependence between different stages as well as correlations between different raypaths or source points (diffraction corners or diffuse scattering points). Therefore, extra computation and communication overhead is unavoidable. It is in this context that we propose techniques that aim to minimize overheads.

4 Considerations for the NOW-based Propagation Prediction Model

There is a wide array of computational paradigms for parallel/distributed computations including the phase-parallel, divide and conquer, pipeline, process farm, and work pool [41]. The phase-parallel paradigm consists of a number of supersteps with each containing a computation phase and an interaction phase. Every processing unit performs an independent computation task in the first phase and executes synchronous communications among all the units in the interaction phase. Since our baseline ray-tracing algorithm functions in stages, it nicely fits the phase-parallel paradigm with the existing constraints of data dependencies between stages. Each stage can be considered as a "super-step" featuring both a computation and an interaction phase. For instance, in the transmitters processing stage, all raypaths emitted by transmitters are decomposed into multiple tasks and each workstation handles a fraction of these tasks in the first phase. During the interaction phase, all processing units exchange intermediate results to generate the needed information for the superstep that computes the diffraction corners and received powers at the receiver locations.

Computations within each superstep are rather complex as there is a close correlation between different raypaths and raypaths do require varying processing times. Thus, it is imperative that computations be partitioned into jobs with fine granularity and be uniformly assigned to different workstations. The above does call for synchronization of all participating sites making inter-processor communication required even during the computation phase. To reduce the inter-processor communication overhead, we adopt the manager/workers paradigm within each stage. One site is appointed the manager and all others are the workers. The manager is responsible for the input processing, computation task distribution, result generations and more importantly for coordination among workers. Each worker repeatedly requests new jobs from the manager, carries out the processing, and ultimately ships back the results to the manager. Evidently, there is no direct communication among workers. As the manager is aware of the current state of all NOW nodes, it is capable of best scheduling and load-balancing tasks. In this regard, the manager makes decisions about workload decomposition and job assignment. Traditionally, two types of decomposition methods have been used: data domain decomposition and control domain decomposition [18]. The former partitions data into non-overlapped subsets assigned by the manager to different sites; the latter divides the original computation into small and disjoint tasks assigned to workstations. Although data domain decomposition is easier to deploy and often features small memory consumption, it has the potential to generate sizable communication overheads.

In our problem domain, a raypath may interact with many buildings. Should these buildings be distributed among various workstations, the tracing of a raypath has to be decided in collaboration with all pertinent sites potentially incurring communication overheads. Data decomposition does not necessarily lead to well balanced NOW-nodes. The distribution of raypaths among different data partitions is in all likelihood not uniform and "hot-spot" data partitions inevitably emerge. Such "hot-spots" which are impossible to determine a priori often contain numerous raypaths that are of interest to multiple workstations rendering some NOW-nodes bottleneck points for the prediction computation. Finally in the data domain decomposition, it is not easy to efficiently determine the termination of a computation as idleness and/or emptiness of job queues at all sites cannot be used as an indicator [9, 19, 75]. In contrast, the control domain decomposition method overcomes the above drawbacks and has a good termination indication (i.e., all sites finish their assigned computations and there is no unassigned task left at the manager). Thus, we advocate control domain as the computation decomposition method. Next, we talk about the specific issues of task granularity, workload balancing schemes, and overall organization of our NOW-based model.

4.1 Task Granularity

In a NOW, the selection of task granularity is crucial as it presents a trade-off between idle nodes and excessive communication costs. The expected NOW speedup is predominantly determined by the site having the maximum finish time. The latter occurs as all nodes complete their ongoing jobs at the same time and the last unprocessed task is awaiting assignment. To decrease this maximum finish time gap, the task granularity should be designated as small as possible while avoiding the potential network message flooding. The minimum feasible task granularity is a single *ray-segment* that typically has processing time at the order of milliseconds. If ray-segment is adopted, the communication overhead will certainly be large and its cost will surpass the original computation cost. Therefore, the task granularity we propose here is either at *raypath-level* or *source-point-level*. If raypath-level task granularity is used, the computation is decomposed based on raypaths (a single raypath or a set of raypaths). In the case of source-point-level task granularity, the computation is divided into tasks based on source points (e.g., transmitters, diffraction corners, or diffuse scattering points). It can also be a single source point, or a set of source points.

The task granularity also depends on the computation stage as the amount of needed computation per stage differs. For example, the number of diffraction corners is often much larger than the number of transmitters [53, 15]. Should the same task granularity be used for different stages, it may lead to lack of load-balancing throughout the stages of the prediction. If the source-point-level task granularity is used for both the transmitters stage and the diffraction corners processing stage when the number of transmitters is less than the number of workstations, a number of sites will remain idle during the entire stage. Similarly, if we use raypath-level task granularity in all stages, the communication overhead for workload assignments in the diffraction corners processing stage will be particularly heavy. Therefore, we use raypath-level task granularity in the transmitters processing stage and source-point-level task granularity during the diffraction corners and diffuse scattering points processing stages.

4.2 Load Balancing Schemes

As time requirements for processing raypaths cannot be determined a-priori and wide variances exist in the processing times for both source points and raypaths, static workload assignment schemes [75] are not suitable for our NOW-based ray-tracing prediction model. Instead, we resort to dynamic load-balancing scheduling schemes, which partition and allocate the workload according to the progress of the computation as well as the state-of-affairs of the NOW-nodes. The latter is a function of past and current status of job execution at nodes as well as the state of the communication substrate. We have considered three load-balancing schemes functioning at the coordinator site in our adopted manager/workers model: fixed-size-task, variable-size-task, and hybrid-size-task schemes [26, 64, 69, 41].

In the fixed-size-task approach, the manager always allocates the same fixed number of computation units G to the requester (worker) no matter who the requester is and when the request is posted. In our model, a computation unit is a raypath during the transmitters processing stage or a source point at the diffraction corners or diffuse scattering points processing stages. The manager does not take into consideration factors such as the current workload, workstation's CPU clock rate, and the behavior of the requester regarding to its past assigned tasks. Should the size of the unassigned computation be T_{rem} computation units, then the fixed-size-task scheme allocates $A = \min(G, T_{rem})$ to the current requester, and updates T_{rem} as $T_{rem} - A$. In the variable-size-task scheme, the manager allocates a variable portion of the remaining workload to the requester. If a NOW consists of N sites, the size of the pending work is T_{rem} and F is an adjustment factor (with $0 < F < 1$ and F experimentally determined), then the size of the current assignment A will be $A = \lceil T_{rem}F/N \rceil$. In the hybrid-size-task scheme, the manager uses the variable-size-task scheme first and checks the size of the current assignment. If it is below a pre-specified threshold G, the manager switches to the fixed-size-task scheme with task size of G. Therefore, the current assignment A to the requester is $A = \min\{\max(\lceil T_{rem}F/N \rceil, G), T_{rem}\}$. Each assignment requires a round-trip message (i.e., a task-request message and a task-assignment message). The performance behavior of the above schemes is analyzed in Appendix A. We derive the condition under which variable-size-task assignment outperforms its fixed-size-task counterpart. In addition, the hybrid-size-task scheme does better than the variable-size-task as long as $G \geq 1$.

4.3 The NOW-Based Radio Propagation Prediction Model

To deploy the baseline Algorithm 1 in a NOW, we assume the existence of a coordinator (or manager) that initially accesses the building/terrain databases and descriptions of transmitters and receivers, accepts user parameters, and delivers the predictions to the user. For the time being, assume that the entire building database can be held in the memory of each workstation, as well as the location of transmitters and receivers and other necessary information [1]. Based on all above discussion, the NOW-based prediction model works as follows:

- Initialization stage: The manager reads the geographic terrain into its main memory, partitions it, and pre-processes all data (e.g., clusters and indexes buildings) that is ultimately broadcasted to all workers in the NOW. The coordinator undertakes the responsibility to read and alert workers about additional user-provided input as well as pertinent configuration information.
- Transmitters processing stage: The manager divides the raypaths from all transmitters into disjoint computation units (tasks or jobs) with raypath-level task granularity. These tasks are scheduled to workers by using a dynamic load-balancing scheme. Since the raypath-level task granularity is used and the processing time for each computation unit (a raypath) is relatively small, large values for F and G can be used to increase the ratio of the computation time over the communication time for each assignment. As soon as all tasks of this phase complete, the manager collects their intermediate results and generates data about the first order of diffractions. The latter is dispatched to all workers.
- Diffraction corners processing stage: This stage is composed of several supersteps, one for each order of diffraction. In each superstep, similar procedure as that used in the transmitters processing stage is followed except that the task granularity is set at source-point-level. To reduce the maximum finish time gap among all workers, small F and G should be used.
- Diffuse scattering points processing stage: All diffuse scattering points are determined and treated as transmitters. The task granularity is set at source-point-level as in this phase we are dealing with diffuse scattering points.

[1] In the next section, we relax this assumption.

- Prediction results generation stage: The manager assembles all intermediate results, compiles predictions, and delivers the results to the user.

If either the problem size is relatively small or the number of NOW nodes is moderate, the manager site can assume a dual responsibility by becoming a concurrent worker as well. On the other hand, if the resources of the coordinator reach saturation, multiple managers can be used to amortize the overall coordination and communication overhead.

5 Distributing Data and Computations in a NOW

5.1 Data Partitioning Techniques

If we were to completely eliminate the communication costs due to data movement, we should maintain an in-memory full copy of the building/terrain databases at every workstation. This would necessitate the highest possible memory consumption of $N \times B$ bytes where N and B are the number of NOW nodes and the size of database respectively. This is an optimistic scenario as in general nodes may have less than B available memory or even the sum of the entire NOW memory may be less than B bytes. By assuming M_i bytes of memory available at the i-th NOW site, we could identify four distinct cases for our data placement problem:

1. *Case I*: $B \leq M_i$ (i = 1, . . ., N) with every site being able to hold the entire building database in main memory (mentioned as baseline case in Section 4).
2. *Case II*: $B \leq M_i$ for some i and $B > M_j$ for some j with i, j assuming distinct values in $[1, N]$. Here, only a subset of workstations can host the entire dataset in memory while the remaining nodes buffer only a fraction of the data.
3. *Case III*: $M_i < B$ (i = 1, . . .,N) and $\sum_{i=1}^{N} M_i \geq B$. No site can store in its own buffer space the entire dataset but the collection of NOW memory available is larger than B.
4. *Case IV*: $M_i < B$ (i = 1, . . ., N) and $\sum_{i=1}^{N} M_i < B$. In this case, no site can hold the entire data in volatile memory and the collective available NOW memory in all sites is less than B.

With the exception of the first case, the trade-offs between communication overheads and memory consumption have to be examined under the assumption that in general accessing data over the network and/or from local disks takes at least an order of magnitude more time than from main memory [41, 18]. In our control domain decomposition method, we move data to appropriate computation sites. To reduce the incurred communication costs, each site should store the largest portion of the building database possible. In an effort to differentiate which portions of the dataset should be core-resident we classify groups of buildings as either "hot-spots" or "cold-spots". We adopt this classification as raypaths are not uniformly distributed among all buildings as limited extent areas often experience heavier concentration of raypaths. In the transmitters processing stage, buildings located nearby transmitter locations intercept many more launched raypaths and are expected to be accessed much more frequently than distant ones designating in this manner hot-spots. As all nodes participate in the processing of transmitter emitted raypaths, it is beneficial if nodes keep in memory buildings located around transmitters during the transmitters processing phase. We can further prioritize buildings placed in cold-spots according to their spatial relationships to formed hot-spots. Such a classification would be beneficial when only part of the cold-spot areas can be memory resident with the remaining cold-spot data stored at the manager's disk.

Building hot-spots can be automatically generated as follows: the manager collects M_i (i = 1, . . ., N) from all workers and determines $B_{hot} = \min_{i=1}^{N} M_i$. The manager computes the number of buildings N_{hot} that can fit in B_{hot}, retrieves the N_{hot} buildings spatially nearest to the transmitters and finally broadcasts these buildings to all workers. Naturally, hot-spots can change during the computation process and different nodes may have different hot-spots especially in the diffraction corners and diffuse scattering points processing stages. Worker i can adjust its hot-spot set of buildings dynamically as follows: every time worker i obtains a new assignment (e.g., a source point), it constructs the hot-spot B'_{hot} for the source point based on M_i such that $B'_{hot} < M_i$ and buildings in B'_{hot} are those nearest to the source point among all buildings. The worker then proceeds to examine whether buildings in B'_{hot} are already memory resident; if not, the worker fetches the new hop-spot elements from the manager at one request that ultimately replace outdated hot-spot items.

Different methods for further classification and allocation of buildings in cold-spots among nodes are adopted for different cases. For example, in *Case III*, the cold-spots may be divided into N parts with each part having size B_i (in bytes for i = 1, . . ., N) so that $B_i \leq (M_i - B_{hot})$ and $\sum_{i=1}^{N} B_i = B - B_{hot}$. Buildings in cold-spots can be assigned to B_i based on their spatial locations. In *Case IV*, the cold-spot buildings are subdivided into two parts, B_{warm} and B_{cold}. The set B_{warm} contains buildings around B_{hot} and consists of N components whose size is B_i with $B_i \leq (M_i - B_{hot})$ for i=1,. . .,N. These N components are delegated for storage to the available main memory of the NOW nodes. All remaining buildings are placed in B_{cold} and are stored on the manager's disk.

For brevity, we discuss the building database partitioning technique for *Case II*—methods for cases *III* and *IV* can be derived in a similar manner. By assuming that the manager's physical memory can accommodate the entire building database, the method works as follows:

- The manager retrieves the entire building database into its memory and collects the sizes of main memory available M_i (i = 1, . . ., N) from all workers. It then generates the hot-spot B_{hot} following the outlined procedure and dispatches it to all other NOW nodes.
- The rest of the building database is sent by the manager to all those workers whose main memory M_i satisfies $B \leq M_i$.
- For sites whose main memory M_j is less than B, the manager delivers buildings in cold-spots upon request. Workers manage and adjust buildings in the hot-spots and cold-spots according to their local situation. Buildings in cold-spots are replaced with a FIFO policy.
- To improve system efficiency, two threads, a computation thread and a communication thread, are used in all sites. The computation thread is in charge of carrying out CPU intensive tasks such as tracing raypaths while the communication thread handles requests for building data from/to other sites.

It is worth pointing out that the above data partition techniques are also used to partition terrain databases and scatter them to machines in a NOW configuration.

5.2 Computation-Duplication and Computation-Partition

Let us assume that computation C consists of two parts C_x and C_y in that order with C_x being much shorter than C_y. In addition, C_x-generated results are not only large but are also heavily used by C_y. Clearly when C is to be distributed, it is advantageous to parallelize C_y as much as possible; should a complete copy of the C_x's results become available at all NOW sites, the entire C would benefit. To obtain a copy of C_x's result at every site, there are two alternatives:

1. *Computation duplication*: C_x is independently repeated by every node, and its results are stored locally for subsequent use by C_y. This method involved little communication and as the requisite time for C_x is very small, it is expected that NOW's performance (especially the speedup) is not affected noticeably.
2. *Computation partition*: C_x is treated just as other parts of computation and each node performs only part of the computation and intermediate results are exchanged among all sites so that each site obtains a complete copy. Although the computation time spent by each site is short (vs. the computation-duplication method), communication overhead is introduced potentially affecting the speedup.

One candidate for such a computation is the establishment of lookup tables. Lookup tables constitute an efficient technique in massive computations problems where values of lookup table are used to avoid time-consuming frequently occurring function calls. In ray-tracing models, the ray-wall-intersection test is the most frequently used operation and it is known to consume up more than 85% of the total processing time [21]. To find out the slope of a line, reflection angle for a ray, or intersection point between a ray and a wall, we need to calculate the tangent or arctangent for a given ray. Performance-wise, it is beneficial to establish a tangent lookup table before computation commences in every NOW node. In Appendix B, we derive processing times for the above two methods; based on our formulae we show that when the number of NOW sites is larger than 2, computation-partition outperforms the computation-duplication method for a 100 Mbps network.

Another candidate for this type of handling is the pre-processing of the building database in order to improve the ray-wall-intersection test operations. This involves partitioning buildings into clusters (or oc-trees) [21, 35, 42], setting up the indexing structure for building retrieval, and creating statistical information about buildings (e.g., size of buildings). In the computation-duplication method, the manager sends the entire building dataset to all sites (it may not be realistic in some cases) and each site may locally pre-process the building database in an independent and simultaneous fashion. In the computation-partition method, all sites take part in the pre-processing procedure but each site just performs a portion of the computation, and exchanges local results with all other sites so that every site gets a complete copy of the results. Similar analysis and computation can be performed as the case of using lookup tables.

In summary, when the number of NOW nodes is at least three and sites are attached to a high bandwidth (at least 100 Mbps), we use the computation-partition method to establish lookup tables and pre-process building/terrain databases. Otherwise, we use the computation-duplication method.

5.3 Intermediate Results Assembly

Due to data and control dependencies among different stages, all input data should be available before a stage commences. In the course of ray-tracing prediction, the output of each stage is information for diffraction corners (or diffuse scattering points) and raypaths illuminating these source points. The corresponding numbers of first and second order diffraction corners as well as raypaths illuminating each corner are huge (see Section 6). Therefore, the output of pertinent stages can be very large and it is a challenge for the manager to collect data from all workers efficiently.

The use of a superstep's interaction phase can assist in the efficient collection of results that are to be consumed by follow-up stages in the prediction computation. To reduce both the complexity of the system and the communication overhead, NOW sites can employ the manager/workers paradigm to materialize the above result assembly in a first-come-first-serve (FCFS) discipline. This method is depicted in Algorithm 2. The manager site is in charge of the assembly operation while all workers are idle before and after they ship out their partial yet locally generated results.

Algorithm 2 First-Come-First-Serve Intermediate Data Assembly Method

```
1: if (the current site is the manager) then
2:   $S \leftarrow D$, where $D$ is the intermediate data at the manager's site, and $S$ is the assembled data
3:   while (there is unprocessed worker) do
4:     receive intermediate data $D$ from the next arriving worker
5:     assemble the received intermediate data $D$ with $S$, that is $S \leftarrow S \cup D$
6:   end while
7: else
8:   send its local intermediate data $D$ to the manager
9: end if
```

To further reduce the idle period each worker may remain at, we propose a multi-level assembly method. A "virtual" binary tree is formed among all NOW sites. Initially, all leaves (sites) are grouped pairwise, that is, sites n_i and n_{i+1} are paired together where $i = 1, \ldots, \lceil N/2 \rceil$. Intermediate data is assembled within each pair of sites and the assembled results are stored at one of them (for example, the site with the lower index). The assembling operation proceeds for each pair simultaneously. Nodes currently holding the assembled results participate in the next round of assembly, while others remain idle. This procedure is repeated until all intermediate data are collected and kept in only one node. To ensure that the manager's site is the final destination, we designate it with identifier n_1 in the above procedure. Algorithm 3 presents our multi-level assembling method with each site having a unique identifier (rank) and the manager assigned-rank equal to one.

Algorithm 3 Multi-Level Intermediate Data Assembly Method

```
1: $f_1 \leftarrow 1$; $f_2 \leftarrow 2$; $S \leftarrow D$, where $S$ is the assembled data, while $D$ is the intermediate data at this site
2: while ($f_1 <$ number_of_nodes(i.e., $N$)) do
3:   if ([(my_rank-1) $modulo$ $f_2$] == 0) then
4:     peer_rank = my_rank + $f_1$;
5:     receive intermediate data $D$ from the site with identifier of peer_rank
6:     $S \leftarrow S \cup D$
7:   else
8:     if ([(my_rank-1) $modulo$ $f_1$] == 0) then
9:       peer_rank = my_rank - $f_1$;
10:      send $S$ to the site with identifier of peer_rank
11:    else
12:      exit;
13:    end if
14:  end if
15:  $f_1 \leftarrow f_2$; $f_2 \leftarrow 2f_2$
16: end while
```

We analytically compare the above two assembly techniques in Appendix C and find that the multi-level assembly method is preferable to FCFS when at least two NOW sites carry out the prediction computation. Hence, the multi-level assembly technique is our choice in our NOW-deployed model.

5.4 Prediction Results Generation

Our radio wave prediction model produces prediction results mainly related to received signal strengths such as system coverage, delay spread, and angle spread observed by receivers. The signal strength reaching a receiver R is determined by all the raypaths illuminating R. To reduce processing time and main memory consumption, we only store a raypath for R if its energy P (expressed in dB) is larger than $P_{max} - P_{thd}$, where P_{max} is the maximum energy among all raypaths illuminating R, and P_{thd} is the pre-specified threshold (typically 20 dB) [7, 53]. A raypath satisfying the above requirement is called a significant raypath. In our model, each site processes disjoint sets of raypaths and stores illuminating raypaths for each receiver locally. To ensure that all raypaths kept at different sites for the same receiver are significant, we prune all non-significant raypaths. We examine two pertinent pruning techniques: "one-site-pruning" and "all-site-pruning". In the former, all workers dispatch their locally stored raypaths to the manager, where the entire pruning takes place. In the "all-site-pruning" technique, each site first determines the local maximum power for each receiver based on its local information. Then, the local maximum power data is exchanged among all sites so that the global maximum power for each receiver can be determined and becomes globally known. Subsequently, each site prunes its own local raypaths for each receiver based on the global maximum powers, and locally generates partial predictions. Finally, all workers send their partial predictions to the manager that is responsible for their synthesis and compilation of final predictions.

The crucial step in "all-site-pruning" is to exchange local maxima and determination of global maxima among all sites. Since we use the MPI standard [22, 23], the global maximum reduction operation (`MPI_Allreduce(·)` with operation code of `MPI_MAX`) can be used to find out the maximum power for each receiver among all sites and the results are delivered to all sites. Similarly, the global summation reduction operation (`MPI_Reduce(·)` with operation code of `MPI_SUM`) can be utilized to perform the assembly of partial results into the final predictions that are stored at one site only (i.e., the manager site). Appendix D compares the performance features of the above two methods and shows that the "all-site-pruning" method has better performance than the "one-site-pruning" method as long as the bandwidth of the network used is high, and the number of sites in the NOW configuration is relatively large. Therefore, we select "all-site-pruning" method to generate final predictions in our model.

5.5 Reduction of Communication Overheads

The communication overhead emanates from information dissemination initiated by the manager to all workers, data movements between different sites when constraints are imposed on the main memory consumption, task allocations, intermediate data collections and final prediction results generation. One way to reduce the communication overhead is to overlap the communication with the computation by using the asynchronous communication mode (also called nonblocking communication or immediate communication mode in [22, 23]). We exploit asynchronous communication mode in the deployment of our prediction model. For instance, in the task allocation procedure, each worker maintains a local task queue holding all unprocessed tasks. When the worker dequeues a task from the task queue, it checks whether or not the task queue is below a specified threshold. If so, it sends a task request to the manager by using the asynchronous communication mode (e.g., `MPI_Isend(·)` function in MPI application programming interface), and returns to its on-going computation immediately. Once the current task is finished, the worker examines whether or not the new task assignment has arrived from the manager by asynchronous communication mode (e.g., `MPI_Wait(·), MPI_Waitany(·)`, or `MPI_Testany(·)` in MPI API). If a new assignment is available, the worker fetches and places it into its task queue.

One drawback of the asynchronous communication mode is its high main memory consumption. The communication substrate needs extra memory to store those messages generated by the asynchronous communication mode, thereby competing with the ray-tracing prediction model for main-memory resources. Another potential drawback is that it may cause some unbalanced workload among sites, which occurs when all workers request new tasks at the same time. If only one unassigned task remains at the manager, it may happen to be sent to a node already involved in a lengthy computation. In this case, the unprocessed tasks accumulate at a single site, while all others complete their own tasks and enter idle periods.

The use of piggybacked messages can substantially reduce communication overhead. In our prediction model, task assignments are carried out by message exchanges between the manager and workers. Task-request and task-assignment messages are often very short, thereby other information can be piggybacked in them. Such information includes non-time-sensitive and non-time-critical data that can tolerate some time delays. They include statistics about computation progress, network status, and some intermediate data (e.g., the maximum power so far for each receiver).

6 Experiments and Evaluations

We have implemented our proposed ray-tracing based radio wave propagation prediction model on a NOW cluster consisting of 26 nodes. All machines are homogeneous Sun Ultra-10 workstations with CPU clock rate of 400 MHz, main memory of 128 MBytes, and swap area of 262 MBytes, inter-networked with a switch at 100 Mbps bandwidth. The nodes run Solaris 5.7 and the communication substrate is provided with Message Passing Interface (MPI) standard [22, 23]. The C and Perl languages have been used in the development of our prototype whose sequential version is available at [7]. Pthreads and their synchronization primitives are used for coordinating concurrent activities within nodes [12]. The main techniques used in each stage are summarized in Table 1.

We have conducted extensive experiments by using both real-life and synthetic building/terrain databases. In all experiments, we assume that only the manager site can access the building databases and configuration files, interact with the user for inputs as well as for predictions display. To simulate constraints on main memory consumption, a user-defined threshold on the usage of main memory may be imposed separately for each site. For the experiments discussed here we have mainly made use of a 2D ray-tracing system which is appropriate for low antennas among tall buildings [8, 73, 7]. When the base station antenna is at a height close to or above the surrounding buildings, it is necessary to consider rays that go over the buildings, as well as around them. Ray-tracing systems handling this case are similar in structure to the 2D ray-tracing systems [53, 58, 7], therefore, the conclusions identified by 2D models are directly applicable to 3D ray-tracing prediction models. The building database used in the 2D ray-tracing system is limited to the footprints of buildings, the heights for all buildings are assumed to be uniform and infinite, and the terrain is assumed to be flat for simplicity. To further corroborate our analysis, we developed a 3D NOW-based

$Stage$	*Specific Techniques used per Stage*
Initialization	building/terrain data accessed by the manager only building/terrain preprocessed with computation duplication or partition lookup tables established by computation duplication or computation partition hot-spots are constructed automatically by the manager
Transmitters processing	manger/workers paradigm and dynamic load balancing scheme raypath-level task granularity multi-level intermediate data assembly method MPI and Pthread for inter-node and intra-node communication asynchronous communication mode for task requests message piggyback to overlap computation and communication
Diffraction corners processing	source-point-level task granularity (diffraction corners) hot-spots are determined by workers locally and automatically (other techniques are similar to “transmitters processing” stage)
Diffuse scattering processing	source-point-level task granularity (diffuse scattering points) hot-spots are determined by workers locally and automatically (other techniques are similar to “transmitters processing” stage)
Prediction results generation	“all-site-pruning” method to prune results computation-partition method to compute final predictions prediction results delivered to user by the manager only

Table 1: Summary of stages and techniques in our proposed NOW-based ray-tracing system

ray-tracing prototype for radio wave propagation prediction based on the Vertical Plane Launching system [7, 53]. In this prototype, 3D terrain databases are used and diffuse scattering is also considered to further improve prediction accuracy.

6.1 Environments used in Experiments

We use four city maps to conduct our experiments. The first map is that of Rosslyn, VA (Figure 3) and contains 79 buildings with 412 walls. Only one transmitter (base station) is treated, located at coordinates (237,656.0, 118,100.0) m. There are 400 receivers (mobile station locations) that are scattered along several streets. The second city map is a synthetic Manhattan-like building area (Figure 4). There are 56 buildings and 224 walls in this map. The single transmitter is located at (340.0, 340.0) m, and 200 receivers are located on two horizontal streets. The third city map we use is that of Turin, Italy shown in Figure 5 and it contains 2,478 buildings with 20,280 vertices. The transmitter is located at (397,910.0, 4,994,980.0) m, and 419 receivers are located around the transmitter. The fourth is the Dupont Circle area in Washington, D.C., shown in Figure 6 which features 3,564 buildings and 23,181 walls. The transmitter is located at (322,780.0, 4,308,550.0) m, while 400 receivers are arranged in two main parallel streets.

The frequency of the carrier used to conduct all experiments is 900 MHz, and the physical phenomena we consider include reflections, diffractions, and diffuse scattering. When 2D building databases are used, the maximum number of reflections each raypath may encounter is 10, and the maximum number of diffractions each raypath may undergo is two (2) when city maps of Rosslyn, Manhattan, and Turin are used, and one (1) when Dupont Circle map is used (for the sake of shortening the experiment time). The corresponding numbers are 4 and 1 when 3D terrain databases are used. The antenna heights used by transmitters (base stations) and receivers (mobile stations) are 10 m and 1.5 m, respectively. All the walls of buildings are assumed to be made of the same material and have dielectric constant $\epsilon_r = 6$. The pincushion ray tracing

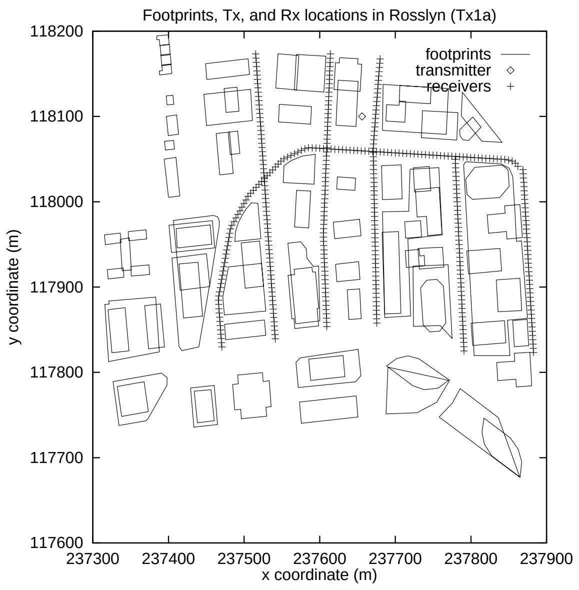


Figure 3: Locations of buildings, a transmitter Tx1a and receivers Rx in Rosslyn, VA

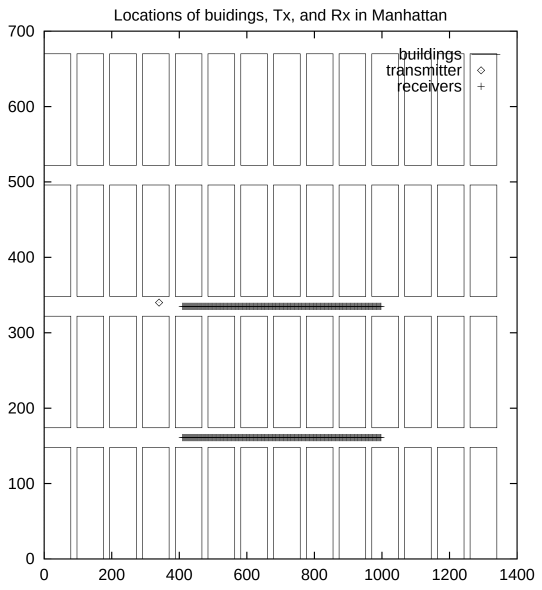


Figure 4: Locations of buildings, Tx and Rx in a Manhattan-like city

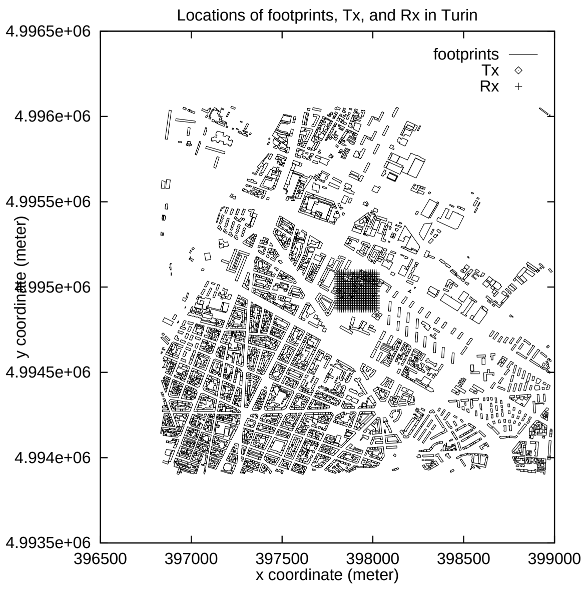
method is used with angular separation of δ = 0.5° unless stated otherwise [7].

To evaluate the performance of our prediction model, we mainly use metrics such as speedup, workload expansion ratio, and resource utilization. Suppose that T_{seq} is the best finish time achieved when only one machine is used, t_i is the finish time for the i-th node when a n-node NOW configuration is used, T_{max} and T_{avg} are the maximum and average finish times, among the n nodes, while T_{sum} is the summation of finish times for all nodes and T_{oh} is the computation and communication overhead, then, $T_{max} = \max_{i=1}^{n} t_i$, $T_{sum} = \sum_{i=1}^{n} t_i$, $T_{avg} = (\sum_{i=1}^{n} t_i)/n$, and $T_{oh} = T_{sum} - T_{seq}$. The speedup S_n, the workload expansion ratio W_n, and the resource utilization U_n can be computed as $S_n = T_{seq}/T_{max} = T_{seq}/\max_{i=1}^{n} t_i$, $W_n = T_{sum}/T_{seq} = \sum_{i=1}^{n} t_i/T_{seq}$, and $U_n = T_{avg}/T_{max} = \sum_{i=1}^{n} t_i/(nT_{max})$. In order to more accurately measure the scalability of our NOW-based model, we employ the metrics of efficiency E_n and isoefficiency K_n whose respective definitions are: $E_n = S_n/n = 1/(1 + T_{oh}/T_{seq})$ and $K_n = E_n/(1 - E_n)$ [49, 30].

6.2 Characteristics of the NOW-based Ray-Tracing System

To investigate the core characteristics of our NOW-based 2D ray-tracing system, we use the city maps for Rosslyn (Figure 3) and Turin (Figure 5) in the following settings: raypath-level and source-point-level task granularities are employed for the processing of transmitters and diffraction corners respectively. In addition, we use the hybrid-size-task dynamic load-balancing scheduling scheme. To compromise between the communication overhead due to task allocation and workload balance among all nodes, the adjustment factor is set to F=1/3 for the transmitter processing stage, while the value F=1/4 is used for the diffraction corners processing stage. To reduce the maximum finish time gap, parameter G is set to two (2) raypaths and two (2) diffractions corners in the transmitters and diffraction corners processing stages respectively. The above parameter values are experimentally calibrated and in this way, we assert the validity of our choices.

We conduct experiments while using up to 26 NOW nodes. For brevity, we present detailed results for up



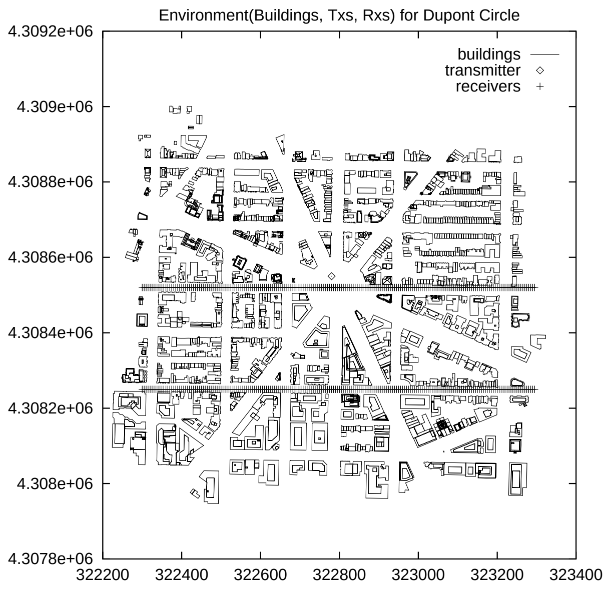


Figure 5: Locations of buildings, Tx (at center) and Rx (around center) in Turin, Italy

Figure 6: Locations of buildings, Tx and Rx in Dupont Circle, Washington D.C.

to 6 nodes and for the case where 26 NOW nodes are in operation. In general, we consistently establish the same trends independent of the number of nodes used. Tables 2 and 3 show the specific statistics obtained when the nodes ranges from 1 to 6 for Rosslyn and Turin respectively. The number of ray trees and finish time (wall clock time) for each node in the transmitter processing stage are listed in columns "*tx-ray*" and "*r-time*". The number of first-order diffraction corners, number of ray trees from these diffraction corners, and finish time (wall clock time) for each node in the diffraction corners processing stage are given in columns "*df-cnr(1)*", "*df-ray(1)*", and "*df-time*". The number of second-order diffraction corners, number of ray trees from these diffraction corners, and finish time for the entire prediction procedure (wall clock time) in each node are listed in columns "*df-cnr(2)*", "*df-ray(2)*", and "*time*" respectively.

Figures 7, 8, and 9 show the results obtained when 26 NOW nodes are employeed for the processing of Rosslyn; similarly for the case of Turin, our results appear in Figures 10, 11, and 12. In all aforementioned graphs, we depict 26 unique identifiers (0 to 25 inclusive) along the *x-axis*–each one of them corresponds to a distinct workstation used in the NOW configuration. Along the *y-axis*, we depict all the related performance measurements as those shown in Tables 2 and 3.

We can observe from Tables 2 and 3 that the number of first-order diffraction corners is large, while the number of second-order diffraction corners is much larger, yet. This is especially true for large databases, such as Turin where the numbers of first-order and second-order diffraction corners are 327 and 10,802, respectively. The first-order diffraction corners are formed due to raypaths originating from the transmitters, while the second-order diffraction corners are generated by rays originating from the first-order diffraction corners. Although the number of diffraction corners depends on both the density of the area and the maximum numbers of reflections and diffractions we consider, the number of diffraction corners appears to exponentially increase with the orders of diffractions. For example in Rosslyn, VA, the numbers of diffraction corners for the first order and second order are 243 and 373, respectively, which are 59% and 91% of the total corners in the entire map (whose number is 412). Here, only about half of the building corners are

# of NOW nodes	*pid*	*tx-ray*	*r-time sec.*	*df-cnr(1)*	*df-ray(1)*	*df-time sec.*	*df-cnr(2)*	*df-ray(2)*	*time sec.*
1	0	720	3.18	243	127704	193.00	373	322079	279.00
2	0	385	1.60	135	71030	96.66	204	178343	139.82
	1	335	1.59	108	56674	96.66	169	143736	139.46
3	0	275	1.42	91	48018	65.62	129	115156	94.72
	1	250	1.42	84	43805	65.62	130	112462	94.22
	2	195	1.42	68	35881	65.51	114	94461	94.54
4	0	185	1.43	60	31286	49.70	103	85295	71.80
	1	90	1.06	68	35062	49.70	96	85049	71.35
	2	245	1.42	57	30027	49.40	85	74201	71.57
	3	200	1.43	58	31329	49.47	89	77534	71.66
5	0	152	1.27	54	28714	39.94	82	71094	57.75
	1	147	1.27	51	26525	39.46	78	67234	57.59
	2	152	1.27	52	27175	39.40	75	66928	57.14
	3	152	1.27	42	21975	39.94	76	61628	57.59
	4	117	1.27	44	23315	39.43	62	55195	57.15
6	0	105	1.22	50	26161	33.79	68	61631	48.65
	1	125	1.22	40	21220	32.76	66	55667	48.65
	2	150	1.22	38	19598	33.79	62	52179	48.60
	3	130	1.22	39	20557	33.10	54	48275	48.37
	4	110	1.22	40	21268	33.23	57	51388	48.43
	5	100	1.22	36	18900	33.48	66	52939	48.30

Table 2: Statistics for Rosslyn, VA when various NOW configurations are used

reached by rays from the transmitter, but almost all building corners are second order diffraction corners.

The processing time for different stages are quite different and the gaps between them are large. We take Turin as an example: the processing time for the only transmitter is 5.00 seconds, while it is 6,775.50 seconds and 31,167.50 seconds for the first-order and second-order diffraction corners, respectively. The average processing time for a first-order diffraction corner and a second-order diffraction corner are quite different. For instance in Rosslyn, the average processing times for a first-order and a second-order diffraction corner are 0.78 seconds [2] and 0.23 seconds respectively. The corresponding values for Turin are 20.71 and 2.89 seconds. It is clear that the processing time for a first-order diffraction corner is much larger than that for a second-order diffraction corner. The main reason is that when tracing the second-order diffraction corners, it is not necessary to check whether rays illuminate any corner, which turns out to be a time-consuming process. The processing time for each individual raypath is quite different as the variance is very large no matter where the raypath comes from (transmitter, first-order diffraction corner, or second-order diffraction corner). For instance, when the NOW configuration has 6 nodes, within the transmitter processing stage in Rosslyn, the number of raypaths processed by node 2 is 150, while only 100 are processed by node 5. The average processing times for raypaths handled by nodes 2 and 5 are 0.0081 seconds (i.e., 1.22/150) and 0.012 seconds (i.e., 1.22/100), respectively. Similarly, under the same NOW configuration, for the first-order diffraction corners in Turin, the numbers of raypaths traced by nodes 2 and 4 are 30,539 and 27,161, while the average processing times are 0.037 seconds and 0.041 seconds respectively. Should we also examine Figures 7, 9, 10, and 12, we can draw the same conclusion. For instance in the case of Rosslyn and during the second-order diffraction corners processing stage, all 26 nodes finish at approximately the same time.

[2] 0.78 seconds = (193.00 - 3.18)/243 seconds

# of NOW nodes	*pid*	*tx-ray*	*r-time sec.*	*df-cnr(1)*	*df-ray(1)*	*df-time sec.*	*df-cnr(2)*	*df-ray(2)*	*time sec.*
1	0	720	5.00	327	174997	6780.50	10802	5870432	37948.00
2	0	385	2.31	165	88586	3475.88	5480	2980156	19196.06
	1	335	2.31	162	86411	3456.81	5322	2890276	19196.08
3	0	385	2.23	108	57961	2274.59	3636	1980445	12660.89
	1	160	2.23	114	60815	2257.05	3772	2046122	12660.90
	2	175	2.18	105	56221	2274.57	3394	1843911	12660.25
4	0	255	2.23	80	43002	1720.38	2758	1498783	9555.60
	1	190	2.11	84	45437	1704.09	2744	1492269	9552.07
	2	140	2.23	85	44819	1703.49	2790	1516543	9553.00
	3	135	2.18	78	41739	1710.38	2510	1363348	9555.59
5	0	197	2.21	64	33983	1350.36	2215	1204257	7555.63
	1	122	2.09	68	36448	1346.05	2172	1185018	7553.18
	2	182	2.08	66	35689	1350.35	2298	1242551	7555.57
	3	102	2.21	63	33627	1347.15	2091	1138679	7553.44
	4	117	2.17	66	35250	1345.24	2026	1100209	7553.79
6	0	125	2.21	52	27812	1139.06	1836	997501	6327.36
	1	95	2.15	58	30451	1121.91	1829	995747	6323.72
	2	110	2.06	57	30539	1139.05	1866	1010467	6327.31
	3	130	2.06	54	29250	1124.96	1861	1012678	6323.66
	4	135	2.06	51	27161	1123.09	1729	939605	6323.84
	5	125	2.06	55	29784	1133.84	1681	914584	6324.40

Table 3: Statistics for Turin when different NOW configurations are used

However, the number of rays emitting from second-order diffraction corners (depicted by curve "*df-ray(2)*" in Figure 9) and processed by each node differs dramatically between a minimum of 9262 (at node 21) and maximum of 16836 (at node 0).

The processing time for each diffraction corner occasionally presents large variance. For example, when 6 NOW-sites are used in the Turin map, the corresponding numbers of second-order diffraction corners processed by nodes 2 and 5 are 1,866 and 1,681. Therefore, the average processing time for these two subsets of diffraction corners are 2.78 seconds [3] and 3.09 seconds respectively. The same observation can be made with the help of Figures 7, 8, 10, and 11. In the case of Turin and during the second-order diffraction corners processing stage, all 26 nodes finish at almost the same time. In contrast, the number of second-order diffraction corners (as shown by curve "*df-cnr(2)*" of Figure 11) processed by each node varies significantly, with minimum of 373 (at node 25) and a maximum of 451 (at node 14).

The maximum finish time gap (difference of finish times among all sites) is very small indicating that workload is well balanced among all sites. More specifically, it is less than 1 second in all experiments for Rosslyn (Table 2), while it is less than 4 seconds in all experiments for Turin (Table 3). This is also evidenced by Figures 7 and 10 where curves representing the finish times of all nodes at different stages are almost horizontal straight lines. In general, the finish time gap is caused by coarse task granularity, which in our experiments is set by G = 2 in assigning diffraction corners. For instance in the Turin case, it can be seen that the average processing time for a second-order diffraction corner, t_{diff2}, is about 3.00 seconds, thereby, the maximum finish time gap is $t_{diff2}G \leq 6$ seconds. It is expected that if we set G = 1, the finish time gap will be further reduced. Tables 2 and 3 also show that the speedup is nearly linear with the number

[3] computed as the fraction (6,327.31 - 1,139.05) / 1,866.

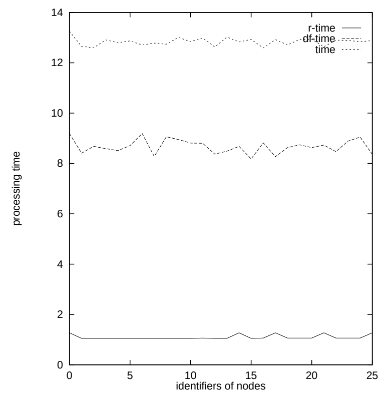


Figure 7: Processing time at various stages of ray-tracing for each node in a 26-node NOW and Rosslyn database

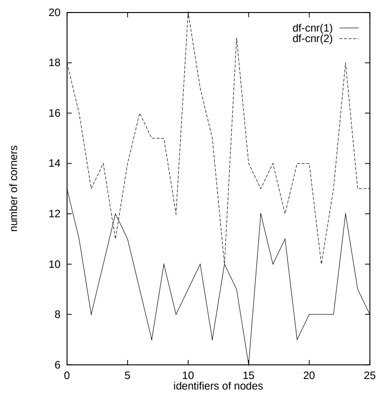


Figure 8: Numbers of corners and rays processed by each node at various stages of ray-tracing in a 26-node NOW and Rosslyn database

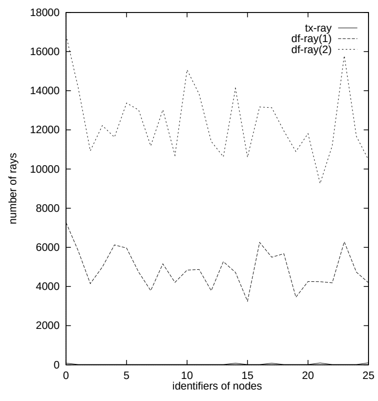


Figure 9: Numbers of corners and rays processed by each node at various stages of ray-tracing in a 26-node NOW and Rosslyn database

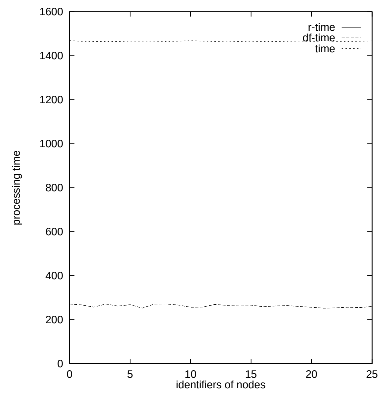


Figure 10: Processing time at various stages of ray-tracing for each node in a 26-node NOW and Turin database

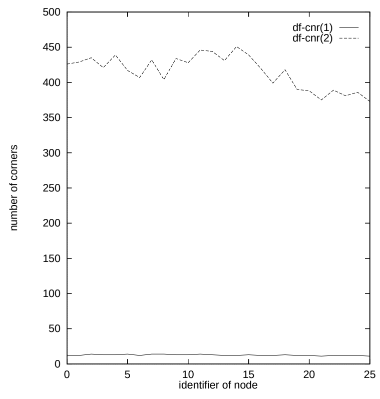


Figure 11: Numbers of corners and rays processed by each node at various stages of ray-tracing in a 26-node NOW and Turin database

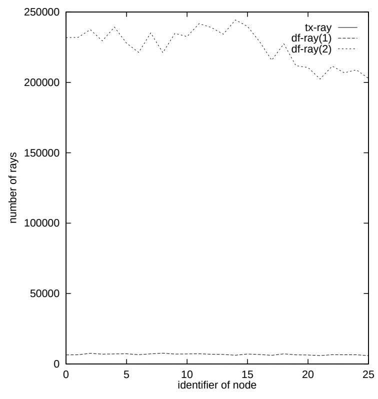


Figure 12: Numbers of corners and rays processed by each node at various stages of ray-tracing in a 26-node NOW and Turin database

of nodes in the NOW (complete speedup presentation can be found in Figure 16).

In summary, the complexity of computations at different processing stages of the ray-tracing procedure is quite different, and the processing times for different raypaths, diffraction corners vary dramatically. However, by using different task granularities for different stages and dynamic load-balancing scheduling scheme, the finish time gaps among nodes can be reduced and linear speedups can be achieved.

6.3 Effects of Task Granularity

To compare the performance of different load-balancing scheduling schemes, we use the city maps of Manhattan (Figure 4) and Dupont Circle (Figure 6) while we vary the number of NOW sites from 1 to 26. We assume that there is no constraint on the amount of main memory a machine can use and the manager ships the entire building database to all workers at the beginning of the computation. Since the hybrid-size-task assignment scheme is better than the variable-size-task assignment scheme (as derived in Appendix A), we show experimental results only for the fixed-size-task scheme and hybrid-size-task scheme. In the fixed-size-task scheme, the size of each assignment G is set to 2 raypaths in transmitter processing stage and 2 diffraction corners in diffraction corners processing stage. In the hybrid-size-task scheme, the adjustment factor F is 1/3 and 1/4 for the transmitter processing stage and the diffraction corners processing stage, respectively. The parameter G is the same as that in the fixed-size-task scheme. Figures 13 and 14 depict the resulting speedup and workload expansion ratios when the fixed-size-task and hybrid-size-task schemes are used for Manhattan and Dupont Circle maps. Tables 4 and 5 show obtained statistics for Manhattan and Dupont Circle that include the minimum (earliest), maximum (latest) finish times among all involved machines (in columns "*min-time*" and "*max-time*"), the summation of finish times for all machines (column "*sum-time*"), and resource utilization (column "*util*").

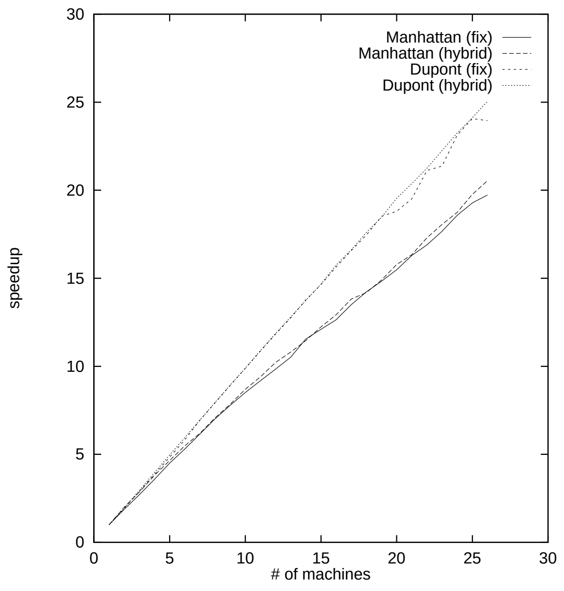


Figure 13: Speedups for Manhattan and Dupont Circle when the fixed-size-task and hybrid-size-task schemes are used

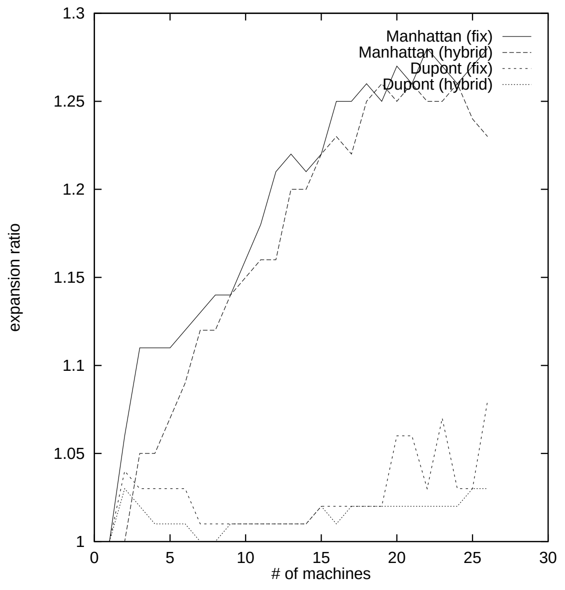


Figure 14: Workload expansion ratios for Manhattan and Dupont Circle when the fixed-size-task and hybrid-size-task schemes are used

Figures 13, 14, and Tables 4, 5 point out that the speedup achieved by the hybrid-size-task scheme is better than its fixed-size-task counterpart in most experiments we have conducted. For instance when the Dupont Circle map is used and the number of sites in NOW is 26, the corresponding speedup rates for the fixed-size-task and the hybrid-size-task scheme are 23.95 and 25.05. Similar observations can be made for other NOW configurations that use the Manhattan building database. It is noteworthy to point out that the relationship between the speedup and the number of sites in the NOW configuration is nearly linear.

The workload expansion ratio is higher for the fixed-size-task scheme than its hybrid-size-task counterpart (Figure 14), indicating that the latter has lower extra computation and communication overhead. For instance when 6 NOW-nodes are involved, the workload expansion ratios for the fixed-size-task and the hybrid-size-task scheme are 1.12 and 1.09 respectively. When the number of NOW sites increases to 26, the corresponding workload expansion ratios are 1.28 and 1.23. The workload expansion ratio is related to the complexity of the computation. In general, it is expected that a more complex computation demonstrates lower workload expansion ratio. Based on experimental results given above, the respective processing times

# of NOW nodes	*min-time sec.*	*max-time sec.*	*sum-time sec.*	*util*	*min-time sec.*	*max-time sec.*	*sum-time sec.*	*util*
	Manhattan (fixed-size-task)				Manhattan (hybrid-size-task)			
1	304.00	304.00	304.00	1.0000	304.00	304.00	304.00	1.0000
2	161.54	161.54	323.08	1.0000	152.37	152.38	304.75	1.0000
4	84.47	84.81	338.64	0.9982	79.50	79.85	318.57	0.9974
6	56.62	57.33	341.25	0.9922	55.09	55.48	331.58	0.9961
8	43.00	43.35	345.55	0.9963	42.54	43.02	341.90	0.9948
10	35.22	35.72	354.07	0.9913	34.79	35.00	348.86	0.9961
12	30.40	30.88	366.91	0.9903	29.18	29.75	353.32	0.9897
14	26.01	26.33	367.00	0.9954	25.81	26.52	364.50	0.9717
16	23.51	24.06	380.74	0.9892	23.08	23.52	373.79	0.9934
18	21.01	21.36	381.74	0.9929	20.94	21.43	381.15	0.9910
20	19.10	19.64	386.26	0.9832	18.84	19.26	381.09	0.9873
22	17.47	17.99	388.47	0.9817	17.14	17.57	381.41	0.9877
24	15.79	16.37	383.78	0.9768	15.75	16.22	382.32	0.9836
26	14.83	15.41	390.00	0.9734	14.23	14.80	373.84	0.9813

Table 4: Statistics for Manhattan when the fixed-size-task and hybrid-size-task schemes are used

for Dupont Circle and Manhattan are 2374.00 and 304.00 seconds when only one node is used to carry out the computation. When 26 nodes take part in the computation, the maximum workload expansion ratios are 1.08 and 1.28, respectively, for Dupont Circle and Manhattan by using the fixed-size-task scheme, while 1.03 and 1.23 by using the hybrid-size-task scheme. It is clear that the workload expansion ratio incurred by the Manhattan map is larger than that of Dupont Circle. The resource utilization rates are similar for both scheduling schemes and very close to the ideal utilization rate indicating that all sites spend little time in idle status.

The maximum finish time gap is very small for both the fixed-size-task and hybrid-size-task schemes. When the Manhattan map is used, the maximum finish gap of 0.71 seconds occurs when the number of nodes in NOW is 6 and the fixed-size-task scheme is used. When the hybrid-size-task scheme is used, the maximum finish gap is also 0.71 seconds, but occurs when the number of sites in the NOW is 14. Similarly, when Dupont Circle map is used, the maximum finish time gaps are 2.04 and 1.98 seconds, respectively, for the fixed-size-task scheme and the hybrid-size-task scheme, both occurring when the number of sites in the NOW is 22. Therefore, the workloads are allocated almost evenly among all sites. Based on the above observations, we conclude that the hybrid-size-task assignment scheme outperforms the fixed-size-task scheme. Figure 15 shows prediction results for transmitter site Tx1a in Rosslyn, VA (as depicted in Figure 3) along with actual measurements obtained in the field readily establishing the accuracy of our model.

6.4 Speedup Rates Without Memory Constraints

Experiments thus far for Manhattan and Dupont Circle using hybrid-size-task scheduling scheme at 26 nodes deliver speedup rates of 20.54 and 25.05, respectively. To evaluate the robustness of our proposed model under varying computation complexities, we also use the city maps for Rosslyn (Figure 3) and Turin (Figure 5) and conduct experiments while ranging the number of NOW-nodes from 1 to 26.

Rosslyn and Manhattan essentially constitute our “light computations” as they feature 56 buildings (with

# of NOW nodes	*min-time sec.*	*max-time sec.*	*sum-time sec.*	*util*	*min-time sec.*	*max-time sec.*	*sum-time sec.*	*util*
	Dupont Circle (fixed-size-task)				Dupont Circle (hybrid-size-task)			
1	2374.00	2374.00	2374.00	1.0000	2374.00	2374.00	2374.00	1.0000
2	1231.20	1231.21	2462.41	1.0000	1219.09	1219.10	2438.19	1.0000
4	613.49	615.11	2457.06	0.9986	593.53	599.54	2396.80	0.9994
6	407.08	407.60	2444.42	0.9995	397.18	398.84	2386.97	0.9975
8	298.46	299.73	2391.92	0.9975	297.84	298.93	2385.72	0.9976
10	238.66	240.13	2395.44	0.9976	238.34	240.15	2390.59	0.9955
12	199.37	200.67	2399.32	0.9964	198.66	200.00	2391.50	0.9965
14	170.94	172.40	2404.51	0.9962	171.26	172.78	2406.15	0.9947
16	150.28	151.81	2417.54	0.9953	149.42	150.72	2401.18	0.9957
18	134.04	136.06	2425.02	0.9902	133.65	134.92	2415.97	0.9948
20	124.83	126.19	2510.47	0.9947	120.19	121.47	2415.90	0.9944
22	110.37	112.41	2440.81	0.9870	109.75	111.73	2426.64	0.9872
24	101.36	102.63	2447.32	0.9936	100.65	102.11	2430.07	0.9916
26	97.84	99.11	2560.34	0.9936	93.22	94.77	2441.35	0.9908

Table 5: Statistics for Dupont Circle when the fixed-size-task and hybrid-size-task schemes are used

224 vertices) and 79 buildings (with 412 vertices) respectively. The higher building density results in longer processing time for Manhattan (304.00 seconds) compared with Rosslyn (279.00 seconds) when a single NOW-node is used. More "intense computations" are those involving Turin (with 2,478 buildings featuring 20,280 vertices) and Dupont Circle (with 3,564 buildings featuring 23,181 vertices). We use two orders of diffractions in Turin as opposed to a single order in Dupont Circle. Therefore, the processing time for Turin is much longer than that of Dupont Circle when a single node is used (37,948.00 and 2,374.00 seconds respectively given in Tables 3 and 5).

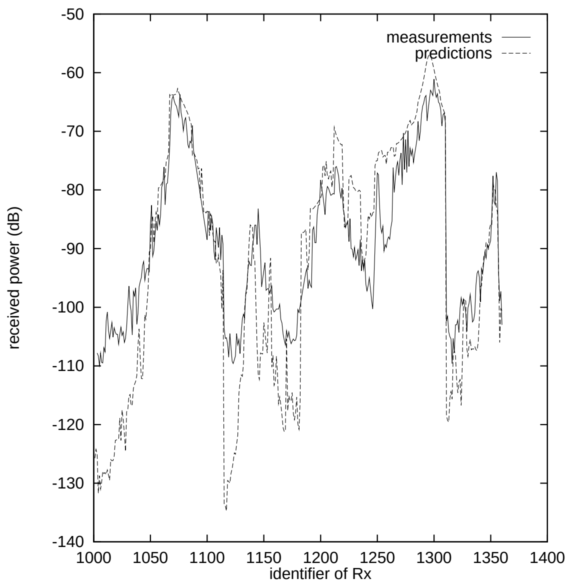


Figure 15: Comparison of predictions for received powers and actual measurements in Rosslyn, VA

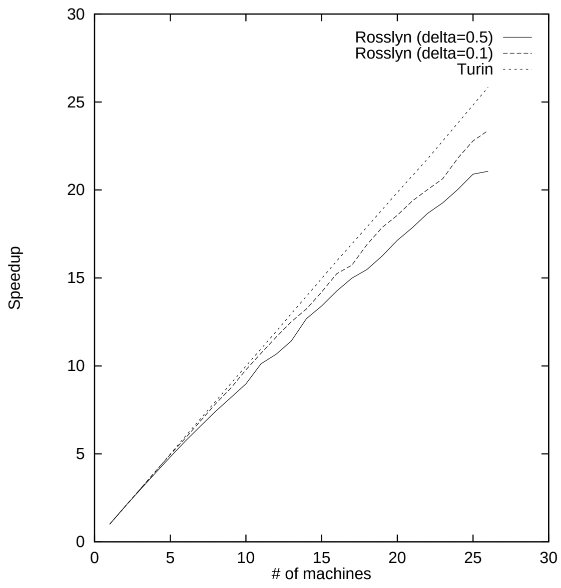


Figure 16: Speedup rates for Rosslyn, VA and Turin, Italy under the hybrid-size-task scheme

We use the same experimental settings as in Section 6.3 and compute statistics including the sum of

# of NOW nodes	*sum-t sec.*	*oh-t sec.*	*iso*	*sum-t sec.*	*oh-t sec.*	*iso*	*sum-t sec.*	*oh-t sec.*	*iso*
	Rosslyn (δ = 0.5°)			Rosslyn (δ = 0.1°)			Turin (hybrid)		
1	279.00	0.00	∞	2350.00	0.00	∞	37948.00	0.00	∞
2	279.28	0.28	435.94	2350.16	0.16	14687.50	38392.14	444.14	85.44
4	286.38	7.38	34.02	2351.01	1.01	225.96	38216.26	268.26	138.29
6	291.00	12.00	21.63	2376.14	26.14	59.67	37950.29	2.29	2348.27
8	298.66	19.66	12.70	2381.15	31.15	50.91	37987.85	39.85	868.77
10	307.26	28.26	8.83	2392.29	42.29	44.01	37980.62	32.62	790.58
12	311.80	32.80	8.04	2398.19	48.19	32.60	38059.16	111.16	287.05
14	305.26	26.26	9.71	2446.10	96.10	17.53	38037.25	89.25	319.70
16	309.58	30.58	8.18	2440.81	90.81	20.05	37967.71	19.71	603.50
18	315.29	36.29	6.15	2455.19	105.19	15.30	38108.93	160.93	187.94
20	317.87	38.87	5.99	2485.11	135.11	12.87	38149.65	201.65	142.98
22	320.36	41.36	5.62	2513.82	163.82	10.10	38307.19	359.19	95.14
24	330.46	51.46	5.04	2513.91	163.91	9.94	38222.58	274.58	119.21
26	333.76	54.76	4.26	2525.12	175.12	8.93	38121.36	173.36	161.07

Table 6: Statistics for Rosslyn and Turin with hybrid-size-task scheme and no memory constraint

finish times (column "*sum-t*" in Table 6), the computation and communication overhead (column "*oh-t*"), isoefficiency (column "*iso*"), speedup (Figure 16), workload expansion ratio (Figure 17), as well as efficiency (Figure 18). The speedup rates are similar for Rosslyn and Manhattan as their processing needs are comparable. The attained speedup for Turin is slightly higher than Dupont Circle; when a 26-node NOW is involved, the corresponding rates for Turin and Dupont Circle are 25.84 and 25.05. This indicates that speedup improves as the computation becomes more complex and the duration of I/O and results delivery phases is small if compared to the parallelizable part. Here, the serial fraction α in Amdahl's law is small yielding a sizable speedup.

The maximum finish time gap tends to increase with the complexity of the computation. The maximum finish time gaps are 0.70, 0.71, 1.98, and 4.15 for Rosslyn, Manhattan, Dupont Circle, and Turin respectively. The workload tends to skew when the complexity of the computation increases. However, the workload expansion ratio decreases with the complexity of computation. For example, when a 26-node NOW is used, the maximum workload expansion ratios are 1.20, 1.26, 1.03, and 1.01 for Rosslyn, Manhattan, Dupont Circle, and Turin, respectively. Therefore, the extra computation and communication overhead (measured by the workload expansion ratio) decreases with the computation complexity as Figure 17 depicts. In addition, Figure 18 shows that the efficiency of our model improves as the computation complexity increases.

Figure 16 clearly shows that the speedup achieved for Rosslyn is less than linear when δ = 0.5°. For instance, when the number of nodes in NOW is 26, the speedup is 21.06 only. Since the computation is relatively small (processing time is 279.00 seconds), when the number of sites is large, the communication overhead becomes significant and slows down the progress of computation. We use Gustafson's law in the Rosslyn context to expand the workload by changing the angular separation δ from 0.5° to 0.1°. The resulting statistics are shown in Table 6. Figure 16 shows that with angular separation δ of $0.1°$, the speedup improves in comparison with that for δ = 0.5°. For example, when 10 NOW sites are present, the speedup rates are 9.78 and 8.98 for δ = 0.1° and 0.5°, while at 26-nodes, the speedup rates are 23.38 and 21.06, respectively.

It is also noticeable that the isoefficiency K_n is much higher for δ = 0.1° than 0.5° (Table 6) indicating

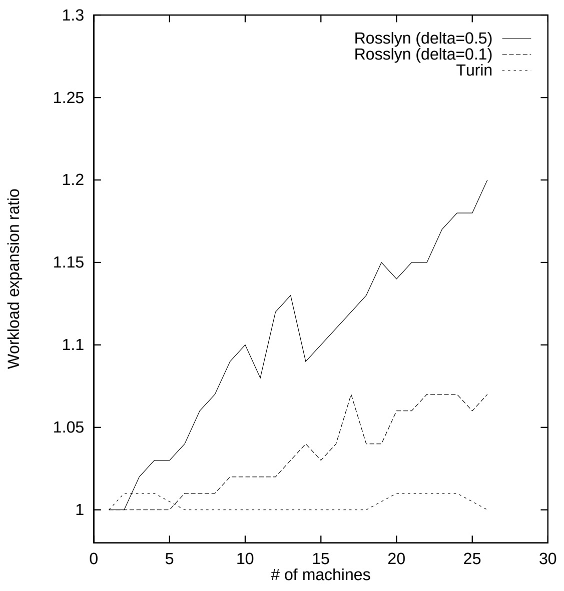


Figure 17: Workload expansion ratios for Rosslyn, VA and Turin, Italy under the hybrid-size-task scheme

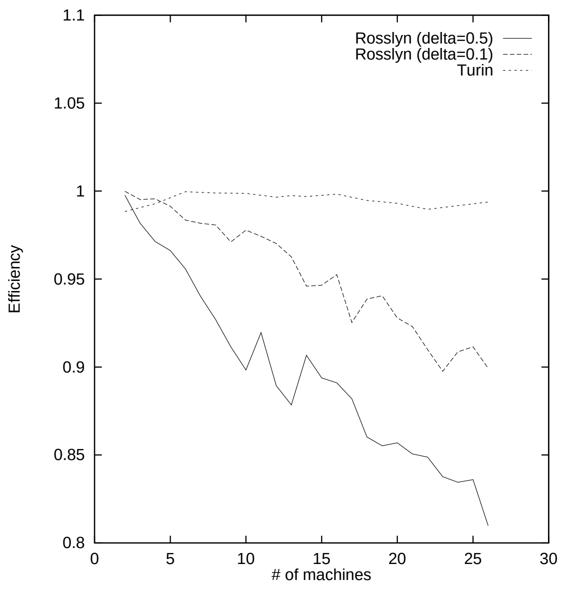


Figure 18: Efficiency (speedup divided by the number of NOW nodes) for Rosslyn and Turin (hybrid scheme)

that the former is more efficient and features better scalability. However, the isoefficiency for both cases decreases rapidly as the number of machines increases in the NOW configuration. It is expected that the isoefficiency may reach zero as the number of sites in the NOW goes up further (e.g., beyond 50). On the other hand, the isoefficiency variation for Turin is not so dramatic raising expectations for better scalability. The same conclusion can also be established with the help of Figure 17. Due to a much lower workload expansion ratio, it is expected that a better scalability will be achieved for Turin.

6.5 Speedup Rates With Memory Constraints

When the entire building database cannot reside in memory, machines can manage database elements using the hot/cold-spot classification discussed in Section 5.1. For brevity, we only offer experimental results pertinent to *Case II* (Section 5.1) that is certain to generate a very large number of message exchanges for fetching building elements. In this context, we discuss two scenarios: in the first, we impose no limit on main memory consumption for NOW machines but with the manager partitioning the building database into two equal parts. The first contains the hot-spots (buildings around the transmitters) while all other buildings make up the cold-spots. The manager sends the first part to all workers at the beginning of the computation, while buildings in the second part are delivered to workers upon request. When a worker fetches buildings in the second part from the manager, it keeps them in its main memory during the remaining computation.

In the second scenario, only the manager can hold the entire building database in its main memory during the entire computation, while all workers can only hold a fraction $i\%$ (i = 97, 95, 90, 85) of the buildings (including both hot-spots and cold-spots) in their main memory. The manager only sends the hot-spot part to all workers at the beginning of the computation, while elements of the cold-spots are delivered to workers on request. If needed, a worker selects a victim building (using FIFO) to accommodate an newly requested/arrived element. For each of the above $i\%$ settings (i.e., i = 100, 97, 95, 90, 85), we vary the number of sites in the NOW configuration from 1 to 26, and run our model for Rosslyn and Dupont Circle. We compute speedup and workload expansion ratio for each execution and present them in Figures 19, 20 for Rosslyn, and Figures 21, 22 for Dupont Circle.

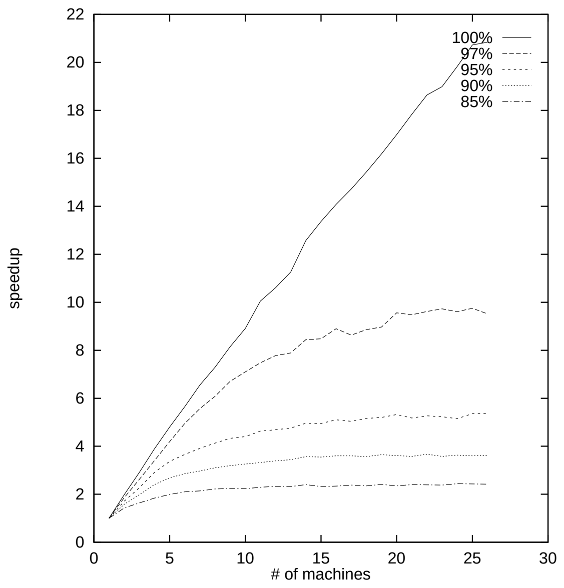


Figure 19: Speedup rates with memory constraints for Rosslyn

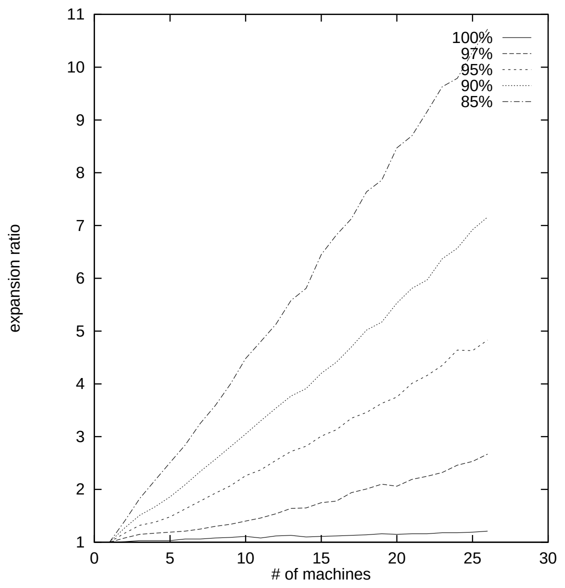


Figure 20: Workload expansion ratios with memory constraints for Rosslyn

Our results show that the effects of main memory constraints on computations are closely related to the complexity of the computations. The lower the complexity, the more the speedup deteriorates. For instance, when each site can only hold 85% of the entire building database in its main memory and the number of nodes in the NOW is 26, the speedups are 2.42 and 23.38, respectively, for Rosslyn and Dupont Circle. The performance for Rosslyn is poor as the net effect of using 26 machines to carry out the computation under this main memory constraint is only equivalent to that of using 3 nodes with no constraint at all. However, for Dupont Circle, the speedup only decreases from 25.05 to 23.38 when the number of nodes in the NOW is 26. Speedup rates proportionally deteriorate when lower percentages of the database were memory-resident at nodes. The main reason for this deterioration is the communication overhead due to fetching of buildings. At some point, the communication overhead dominates the computation and becomes the bottleneck of the system, as can be seen from the change of the workload expansion ratio. We use Rosslyn under the main memory constraint of 90% as an example. When the number of nodes in the NOW is less than 6, the workload expansion ratio is about 2, which is equivalent to solving a problem twice the size of the original one. Similarly, when the number of nodes in the NOW is between 16 and 22, the workload expansion ratio is about 5, it is expected that its speedup is only about 1/5 of that without main memory constraint.

In the first scenario outlined above (curves under “100%” in Figures 19, 21), speedup rates are also affected noticeably for all city maps. For example, under the 26-node NOW configuration, the speedups are 20.84 and 24.74, respectively, for Rosslyn and Dupont Circle, while in the case where the manager sends the entire building database to all workers at the beginning of the computation, the speedups are 21.06 and 25.05, respectively (see Figures 16 and 13). Overall, if main-memory is limited, workers should

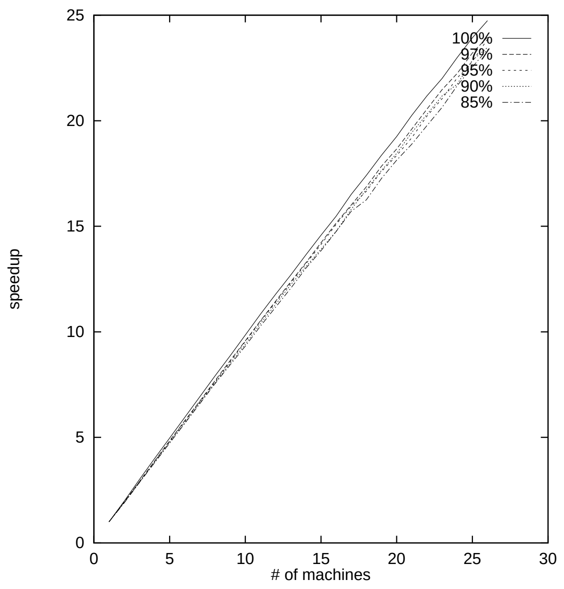


Figure 21: Speedup rates with memory constraints for Dupont Circle

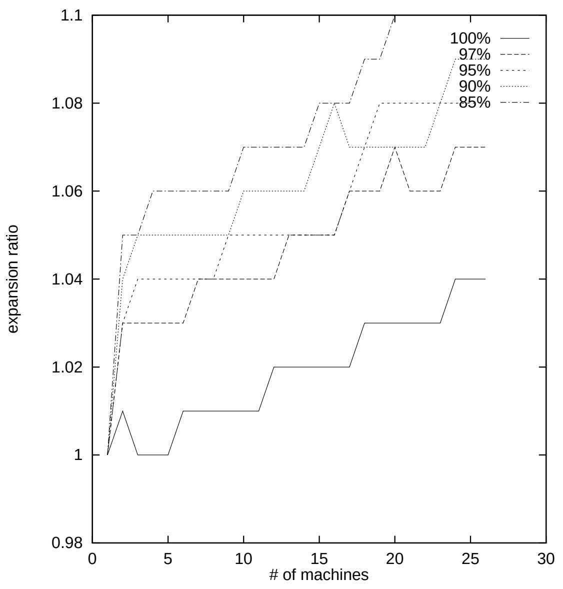


Figure 22: Workload expansion ratios with memory constraints for Dupont Circle

accommodate hot-spots in memory while fetching other database elements on demand in order to reduce communication overhead. In addition, if the complexity of a given computation is low in the presence of limited memory, it is not beneficial to use a large-sized NOW configuration.

6.6 Enhancements With Diffuse Scattering

The diffuse scattering effect is a critical physical phenomenon in radio wave propagation. Here, we classify building surfaces into two types, namely Lambertian and reflection surfaces. A surface is Lambertian if it can establish a line-of-sight (LOS) path and is within a pre-specified distance (e.g, 30 m in the following experiments) to the base or mobile stations. Otherwise, a surface is considered a reflection surface. If a raypath hits a reflection surface, only a reflection path is generated. However, if a raypath intersects a Lambertian surface, the latter re-emits rays uniformly in all directions along with a reflection ray [14, 7]. We only consider the receiver side diffuse scattering. Each Lambertian surface is partitioned into small meshes (in the following experiments, the size of a mesh is $3\times 3\,\text{m}^2$) and the centroid of the mesh is used as the diffuse scattering point that is traced as if it were a transmitter.

We investigate the impact of diffuse scattering in the context of 3D ray-tracing using a 3D terrain database and compare performance rates with those derived for 2D ray-tracing. To accomplish this, we use the building database for Rosslyn, VA (see Figure 3). Figure 27 depicts its building height distribution with maximum and average values at 99.62 m, and 29.37 m; this is a typical high-core city with most buildings ranging from 3 to about 25 stories. Base stations and antennas with different heights are placed in a number of locations (see columns “ID for base stations”, “locations”, and “antenna” of Table 7). To create realistic-terrain conditions in the database, we partition the coverage of the wireless network into a grid of small uniform-sized cells (e.g., with size of $10\times 10\,\text{m}^2$) and then for each such cell we randomly generate this cell’s terrain height in the range of [0, 5] m. All the buildings and receiver locations within this grid are recomputed based on the terrain information; the recomputed dataset is the entry to our NOW-based radio wave propagation prediction model. For brevity, we trace each raypath upto 4 reflections and 1 diffraction.

IDs for base stations	*locations (x, y, z) in (m,m,m)*	*antenna m*	*2D sec.*	*3D (no diffuse) sec.*	*3D (diffuse) sec.*
Tx1a	(237,656.0, 118,100.0, 21.63)	10	279.00	1726.29	11562.11
Tx5	(237,621.0, 117,816.0, 72.00)	2	261.55	1569.30	7610.34
Tx6	(237,518.0, 117,952.0, 73.20)	2	271.68	1793.10	7947.76
Tx4a	(237,655.0, 117,998.0, 23.30)	5	409.31	2619.65	16372.79
Tx4b	(237,655.0, 117,998.0, 23.30)	10	300.61	2104.27	13361.18
Tx10	(237,567.0, 117,737.0, 37.00)	10	250.33	1627.15	9491.62

Table 7: Prediction times for 2D/3D Rosslyn, VA database (with or without diffuse scattering effect)

While retaining identical parameter settings with those of Section 6.3, we first conduct experiments with a single machine and compile the results appearing in Table 7 for a) 2D ray-trace only; b) 3D ray-trace without diffuse scattering effect; and, c) 3D ray-trace with diffuse scattering effect. When base station Tx1a is used, the prediction results generated by cases (a) and (b) along with the measurements obtained from the field are shown in Figure 23. Predictions generated by cases (b) and (c) are depicted in Figure 24. We also provide the prediction results (and corresponding actual measurements) for cases (b) and (c) when base stations Tx4b and Tx10 (with their locations shown in Figure 7) are used in Figures 25 and 26 respectively.

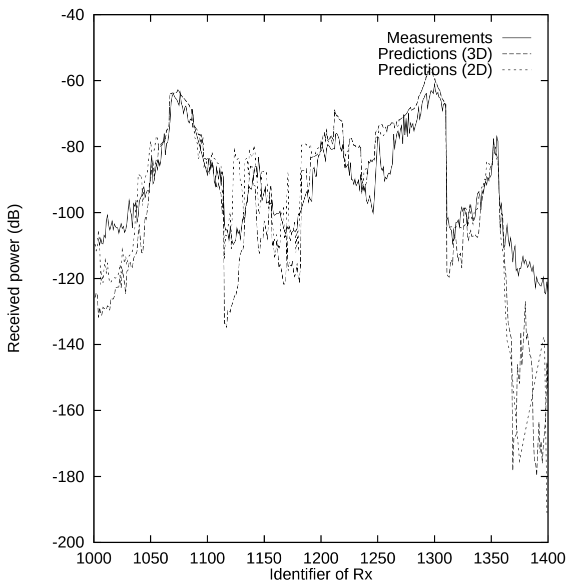


Figure 23: Comparison of measurements and predictions without diffuse scattering with 2D and 3D building databases (Rosslyn, Tx1a)

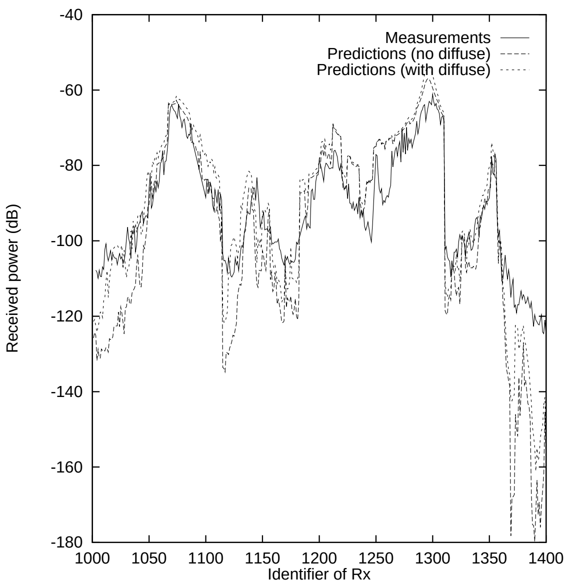


Figure 24: Comparison of measurements and predictions with and without diffuse scattering with 3D building database (Rosslyn, Tx1a)

Next, we conduct the same experiments on a NOW whose number of nodes vary from 1 to 26. Figure 28 shows the obtained speedup rates for all three settings when base station Tx1a is used. Results in Table 7 clearly point out that, when a 3D ray-trace is used, prediction times increase significantly; this increase becomes even more notable when diffuse scattering is taken into account. For example, when base station Tx1a and 3D ray-tracing are used, the required processing times are 1,726.29 seconds and 11,562.11 seconds

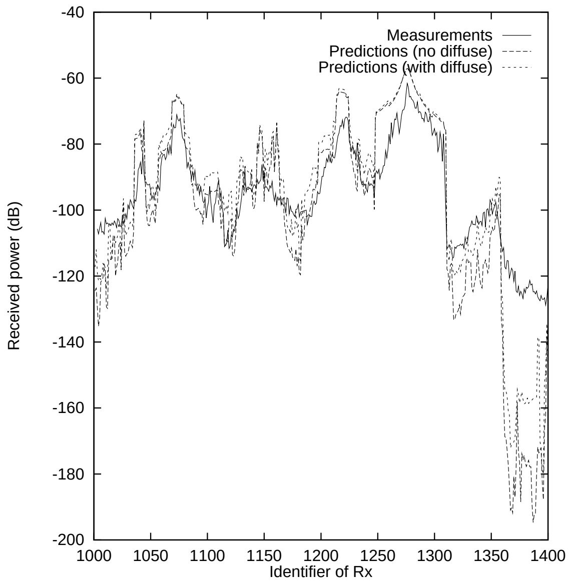


Figure 25: Comparison of measurements and prediction with/without diffuse scattering in Rosslyn, VA for base station Tx4b

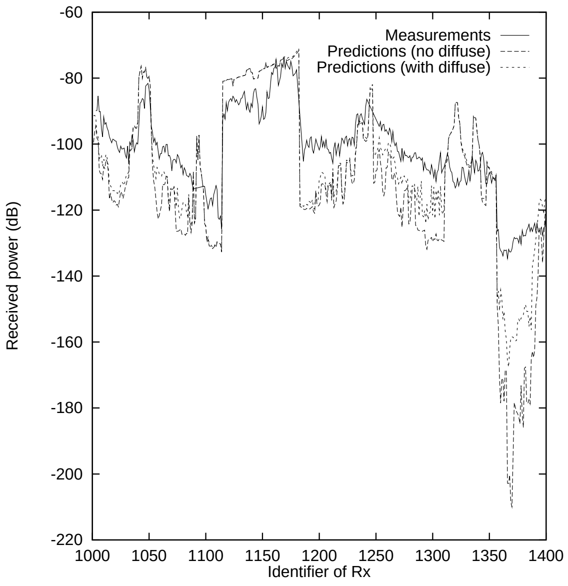


Figure 26: Comparison of measurements and prediction with/without diffuse scattering in Rosslyn, VA for base station Tx10

without and with diffuse scattering effect. These values are 6.19 and 41.44 times higher if compared to that needed to process the 2D setting. This is the reason that the curve corresponding to diffuse scattering in Figure 28 demonstrates the best overall speedup.

The accuracy of prediction results is generally, but not necessarily better for 3D ray-tracing if compared with those derived with 2D ray-tracing. In the latter, the height of buildings is assumed to be infinity, thereby causing reflection of raypaths that in 3D would pass over the tops of buildings. This effect can cause the 2D predictions to be higher than those obtained in 3D in some locations, and lower in other locations.

Diffuse scattering improves the prediction accuracy significantly as shown in Figures 24–26. The improvement of prediction accuracy is at the cost of prediction time. The workload is much heavier when we consider diffuse scattering. A larger workload helps to improve the speedup rates as indicated in Figure 28. It is clear that the speedup achieved by using 3D building database is better than that of the 2D dataset, while it is much better when diffuse scattering effect is taken into account.

7 Conclusions and Future Work

In this paper, we address the computationally expensive problem of radio wave propagation and we propose a NOW-based ray-tracing model to overcome long response times in attaining accurate prediction results. Our model combines both phase parallel and manager/workers paradigms to offer scalable performance. As the ray-tracing process is carried out in stages due to inherent data and control dependencies, we employ the phase parallel paradigm to provide coordination between computational stages. The phase parallel paradigm presents the best match to the characteristics of our overall radio wave propagation prediction process. The manager/workers paradigm is used within each stage to control, coordinate, and synchronize the

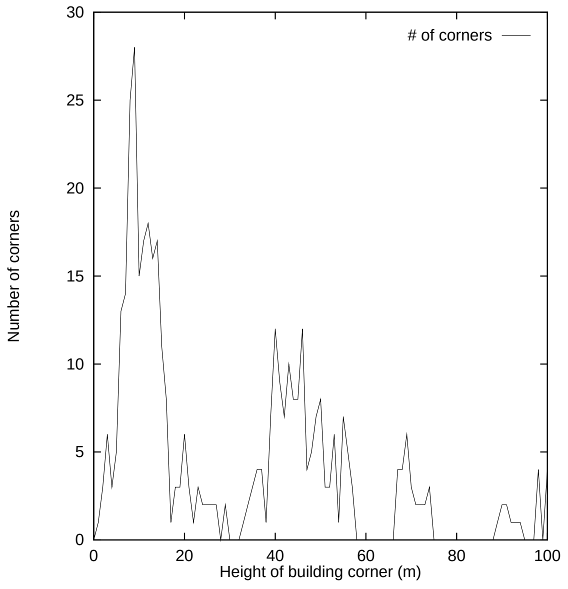


Figure 27: Distribution of building heights in Rosslyn, VA

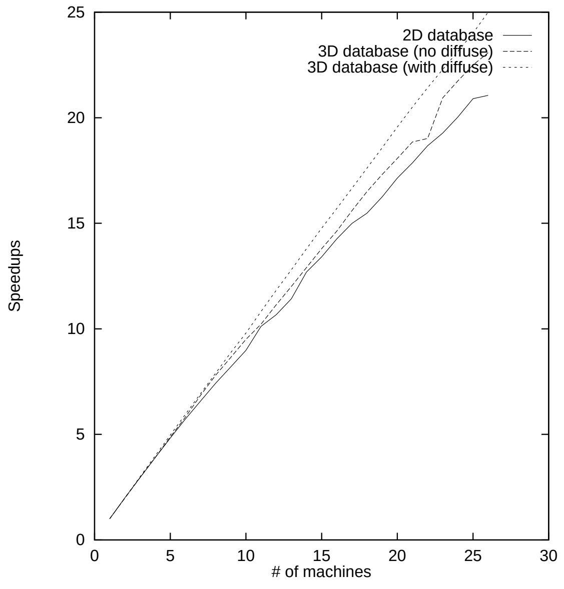


Figure 28: Speedup rates with 2D and 3D building databases for Rosslyn, VA

computation and communication among different nodes. To decompose a computation into small units, we use the control domain decomposition method that allows for the partition of a computation based on task granularity. In particular, we use raypath-level task granularity for transmitter processing and source-point-level task granularity for diffraction corner and diffuse scattering point processing. To allocate workload uniformly among all sites, we utilize a dynamic hybrid-size-task scheduling scheme that takes into account both the state-of-affairs at NOW-nodes and progress of the on-going computation. To further improve the efficiency of our model, we investigate and design suitable techniques for intermediate results collection, final predictions generation, and reduction of extra computation and communication overhead.

Main memory consumption is a crucial issue when control domain decomposition methods are used to partition the problem in consideration. When the building database is large, it can be automatically partitioned into hot-spot and cold-spot parts. The hot-spot consists of those buildings around the transmitters, diffraction corners, or diffuse scattering points, while other buildings are designated as the non-hot-spot (cold-spot) area. During the entire computation, workers always store hot-spot part in their main memory, while other non-hot-spot buildings are fetched from the manager on demand, and may be replaced by other newly fetched buildings based on a replacement policy (e.g., FIFO).

Experiments with our NOW prototype show that when no constraint is imposed on the main memory consumption, the proposed prediction model can achieve speedup nearly-linear to the number of participating nodes. Our model is robust under computations featuring different complexity and processing time requirements. The workload expansion ratio increases slowly along with the number of nodes in the NOW configuration indicating that our model has reasonable computation/communication overheads and good scalability. The scalability of the proposed NOW-based model can be further verified by its isoefficiency metric. The resource utilization is close to the ideal value, implying that all nodes spend little time in the idle state during the entire computation process. When main memory consumption is a concern, the proposed prediction model still delivers very good performance when the duration of undertaken computations is long.

In this setting, the introduced extra computation and communication overheads do not dominate the original computation. Our experimental results also show that the speedup rates can be significantly improved when 3D building/terrain databases are used. Finally, when diffuse scattering is considered, our model not only offers predictions closer to actual measurements but also further improves the obtained speedup rates.

We plan to extend our work in the field of radio wave propagation prediction by pursuing a number of issues: first, we intend to investigate the deployment of our model in a heterogeneous networked environment (not necessarily a NOW) where contributing nodes might display varying characteristics; second, further examine the performance of our prototype with wide-area GIS databases; and finally, study the viability of our model in modern computational environments where resources may join and/or depart the on-going computations in a dynamic manner.

Acknowledgments: We are grateful to the reviewers for their detailed comments that helped us significantly improve the presentation of our work. We are also very much indebted to Dr. Boris Aronov and Dr. Yi-Jen Chiang for reviewing earlier versions of the manuscript.

References

[1] W. Ahnert. EARS: Auralization Software. *Journal of the Audio Engineering Society*, 11(41):894–904, November 1993.

[2] S. M. Akramullah, I. Ahmad, and M. L. Liou. A Data-Parallel Approach for Real-time MPEG-2 Video Encoding. *Journal of Parallel and Distributed Computing*, 30(2):129–146, November 1995.

[3] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin/Commings Publishing, Menlo Park, CA, 2nd edition, 1994.

[4] G. M. Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS 1967 Spring Joint Computer Conference*, volume 40, pages 483–485, 1967.

[5] Y. Ando. *Concert Hall Acoustics*. Springer-Verlag, New York, NY, 1985.

[6] L. L. Beranek. *Concert and Opera Halls: How They Sound*. Acoustical Society of America, New York, NY, April 1996.

[7] H. L. Bertoni. *Radio Propagation for Modern Wireless Systems*. Prentice-Hall PTR, Upper Saddle River, NJ, 2000.

[8] H.L. Bertoni, W. Honcharenko, L.R. Maciel, and H.H. Xia. UHF Propagation Prediction for Wireless Personal Communications. *Proceedings of the IEEE*, 82(9):1333–1359, September 1994.

[9] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation Numerical Methods*. IEEE Computer Society Press, Los Alamitos, CA, 1989.

[10] G. Blelloch and G. Narlikar. A Practical Comparison of N-body Algorithms. In S. N. Bhatt, editor, *Parallel Algorithms*, volume 30 of *Series in Discrete Mathematics and Theoretical Computer Science*, pages 81–96. American Mathematical Society, 1997.

[11] G. E. Blelloch. Programming Parallel Algorithms. *Communications of the ACM*, 39(3):85–97, March 1996.

[12] D. Butenhof. *Programming with POSIX Threads*. Addison Wesley, Reading, MA, 1997.

[13] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen. The SIMNET Virtual World Architecture. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, pages 450–455, Seattle, WA, September 1993.

[14] Z. Chen, H. L. Bertoni, and A. Delis. Comparison of Lambert's Law and Keller Cone Scattering on Urban Propagation Predictions. *Technical Report, Polytechnic University, Brooklyn, NY*, 2004.

[15] Z. Chen, H. L. Bertoni, and A. Delis. Progressive and Approximate Techniques in Ray-Tracing Based Radio Wave Propagation Prediction Models. *IEEE Transactions on Antennas and Propagation*, 52(1):240–251, January 2004.

[16] Z. Chen, A. Delis, and H. L. Bertoni. Building Footprint Simplification Techniques and Their Effects on Radio Propagation Predictions. *The Computer Journal*, 47(1):103–133, January 2004.

[17] D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture – A Hardware/Software Approach*. Morgan Kaufmann Publishers, Inc., San Fransisco, CA, 1996.

[18] A. Delis, V. Kanitkar, and G. Kollios. Database Architectures. In John Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*. John Wiley and Sons, New York, NY, February 1999.

[19] E. W. Dijkstra, W. H. Feijen, and A. J. M. V. Gasteren. Derivation of a Termination Detection Algorithm for a Distributed Computation. *Information Processing Letters*, 16(5):217–219, 1983.

[20] N. I. Durlach and A. S. Mavor. *Virtual Reality Scientific and Technological Challenges*. National Academy Press, Washington, D.C., 1995.

[21] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics – Principles and Practice*. Addison-Wesley, Reading, MA, 1990.

[22] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. June 1995.

[23] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface. July 1997.

[24] S. H. Foster, E. M. Wenzel, and R. M. Taylor. Real-Time Synthesis of Complex Acoustic Environment. In *Proceedings of the ASSP(IEEE) Workshop on Application of Signal Processing to Audio and Acoustics*, Mohonk, New Paltz, NY, October 1991.

[25] G. Fox, M. Jonhson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[26] B. Freisleben, D. Hartmann, and T. Kielmann. Parallel Raytracing: A Case Study on Partitioning and Scheduling on Workstation Clusters. In *Proceedings of Thirtieth International Conference on System Sciences*, pages 596–605, Maui, HI, 1997.

[27] B. Freisleben, D. Hartmann, and T. Kielmann. Parallel Incremented Raytracing of Animations on a Network of Workstations. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1305–1312, Las Vegas, NV, July 1998.

[28] T. Funkhouser, J. M. Jot, and N. Tsingos. Computational Sound for Graphics, Virtual Reality, and Interactive Systems. In *Proceedings of SIGGRAPH 2002*, San Antonio, TX, July 2002.

[29] T. Funkhouser, P. Min, and I. Carlbom. Real-Time Acoustic Modeling for Distributed Virtual Environments. In *Proceedings of the ACM Computer Graphics Conference (SIGGRAPH'99)*, pages 365–374, Los Angeles, CA, August 1999.

[30] A. Grama, A. Gupta, and V. Kumar. Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures. *IEEE Parallel and Distributed Technology: Systems and Applications*, 1(3):12–21, 1993.

[31] A. Greenbaum. Parallelizing the Adaptive Fast Multipole Method on a Shared Memory MIMD Machine. *Ultra-computer Note 162*, June 1989.

[32] L. J. Greenstein, J. Bach Andersen, H. L. Bertoni, S. Kozono, and D. G. Michelson. Channel and Propagation Models for Wireless System Design I. *IEEE Journal on Selected Areas in Communications*, 20(3):493–495, April 2002.

[33] J. L. Gustafson. Reevaluating Amdahl's Law. *Comm. of ACM*, 31(5):532–533, 1988.

[34] J. He, A. Verstak, L. T. Watson, T. S. Rappaport, C. R. Anderson, N. Ramakrishnan, C. A. Shaffer, W. H. Tranter, K. Bae, and J. Jiang. Global Optimization of Transmitter Placement in Wireless Communication Systems. In A. Tentner, editor, *Proc. High Performance Computing Symposium 2002*. Soc. for Modeling and Simulation International, San Diego, CA, 2002.

[35] D. Hearn and M. P. Baker. *Computer Graphics – C version*. Prentice Hall, Englewood Cliffs, NJ, 1997.

[36] Y. Hu and S. L. Johnsson. A Data-parallel Implementation of Hierarchical N-body Methods. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(1):30–40, 1996.

[37] Y. Hu, S. L. Johnsson, and S. H. Teng. High Performance Fortran for Highly Irregular Problems. In *Proceedings of the 6th ACM Symp. on Principles and Practice of Parallel Programming*, ACM press, 1997.

[38] P. Huttunen, J. Ikonen, J. Porras, and K. Sipila. Parallelization of Propagation Model Simulation. In *Science Technology: Science and Art. 10th European Simulation Symposium'98*, Nottingham, United Kingdom, October 1998.

[39] P. Huttunen, J. Porras, J. Ikonen, and K. Sipila. Using Cray T3E for the Parallel Calculation of Cellular Radio Coverage. In *Proceedings of the Eurosim'98*, pages 27–32, Helsinki, Finland, April 1998.

[40] K. Hwang. *Advanced Computer Architecture with Parallel Programming*. McGraw-Hill, New York, NY, 1993.

[41] K. Hwang and Z. Xu. *Scalable Parallel Computing – Technology, Architecture, Programming*. McGraw-Hill, New York, NY, 1998.

[42] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw Hill International Editions, Singapore, 1995.

[43] V. Kanitkar and A. Delis. Site Selection for Real-Time Client Request Handling. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, Austin, TX, USA, May/June 1999.

[44] T. Kawai. Sound Diffraction by a Many Sided Barrier or Pillar. *Journal of Sound and Vibrations*, 79(2), 1981.

[45] H. J. Kim and C. M. Kyung. A New Parallel Ray-Tracing System Based on Object Decomposition. *The Visual Computer*, 12(5):244–253, June 1996.

[46] S.C. Kim, B.J. Carino, T.M. Willis, V. Erceg, S.J. Fortune, R.A. Valenzuela, L.W. Thomas, J. Ling, and J.D. Moore. Radio Propagation Measurements and Prediction Using Three-Dimensional Ray Tracing in Urban Environments at 908MHz and 1.9GHz. *IEEE Transactions on Vehicular Technology*, 48(3):931–946, May 1999.

[47] S. Krishnan and L. V. Kale. A Parallel Adaptive Fast Multipole Algorithm for N-body Problems. In *Proceedings of the 24th International Conference on Parallel Processing*, pages III:46–51, Oconomowoc, WI, August 1995.

[48] U. R. Krockstadt. Calculating the Acoustical Room Response by the Use of a Ray Tracing Technique. *Journal of Sound and Vibrations*, 8(18), 1968.

[49] V. Kumar and A. Gupta. Analyzing Scalability of Parallel Algorithms and Architectures. *Journal of Parallel and Distributed Computing*, 22(3):379–391, 1994.

[50] T. Kurner and A. Meier. Prediction of Outdoor and Outdoor-to-Indoor Coverage in Urban Areas at 1.8 GHz. *IEEE Journal on Selected Areas in Communications*, 20(3):496–506, April 2002.

[51] H. Kuttruff. *Room Acoustics*. Elsevier Applied Science, New York, NY, 1991.

[52] K. H. Kuttruff. Auralization of Impulse Responses Modeled on the Basis of Ray-Tracing Results. *Journal of the Audio Engineering Society*, 41(11):876–880, November 1993.

[53] G. Liang and H. L. Bertoni. A New Approach to 3-D Ray Tracing for Propagation Prediction in Cities. *IEEE Transactions on Antennas and Propagation*, 46:853–863, 1998.

[54] P. Liu and S. N. Bhatt. Experiences with Parallel N-body Simulations. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'94)*, pages 122–131, Cape May, NJ, June 1994.

[55] M. J. Muuss. Towards Real-Time Ray-Tracing of Combinatorial Solid Geometric Models. In *Proc. BRL-CAD Symposium'95*, Aberdeen Proving Grounds, MD, June 1995.

[56] M. J. Muuss and M. Lorenzo. High Resolution Interactive Multispectral Missile Sensor Simulation for ATR and DIS. In *Proc. BRL-CAD Symposium'95*, Aberdeen Proving Grounds, MD, June 1995.

[57] L. S. Nyland, J.F. Prins, and J. H. Reif. A Data-parallel Implementation of the Adaptive Fast Multipole Algorithm. In *Proceedings of the 1993 DAGS/PC Symposium*, pages 111–123, Hanover, NH, June 1993.

[58] W.M. Obrien, E.M. Kenny, and P.J. Cullen. An Efficient Implementation of a Three-Dimensional Microcell Propagation Tool for Indoor and Outdoor Urban Environments. *IEEE Transactions on Vehicular Technology*, 49(2):622–630, March 2000.

[59] J. J. Ottusch, M. A. Stalzer, J. L. Visher, and S. M. Wandzura. Scalable Electromagnetic Scattering Calculations on the SGI Origin 2000. In *Proceedings of the ACM/IEEE Supercomputing'99 Conference*, pages 1–12, Portland, Oregon, November 1999.

[60] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. P. Sloan. Interactive Ray Tracing. *Symposium on Interactive 3D Graphics(I3D)*, pages 119–126, April 1999.

[61] A. D. Pierce. *Acoustics – An Introduction to Its Physical Principles and Applications*. American Institute of Physics, 1984.

[62] S. Ranka and Sartaj Sahni. *Hypercube Algorithms with Applications to Image Processing and Pattern Recognition*. Springer-Verlag, New York, 1990.

[63] E. Reinhard. A Parallelization of Ray Tracing with Diffuse Interreflection. In *Proceedings ASCI 1996 Conference*, pages 367–372, Belgium, June 1996.

[64] E. Reinhard and A. Chalmers. Message Handling in Parallel Radiance. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface (editors: M. Bubak, M. Dongarra, J. Wasniewski)*, pages 486–493. Springer-Verlag, November 1997.

[65] A. Reisman, C. Gotsman, and A. Schuster. Interactive-Rate Animation Generation by Parallel Progressive Ray-Tracing on Distributed Memory Machines. *Journal of Parallel and Distributed Computing*, 60(9):1074–1102, 2000.

[66] I. Rigoutsos and A. Delis. Managing Statistical Behavior of Large Data Sets in Shared-Nothing Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 9(11):1073–1088, November 1998.

[67] D. Robertson, K. Campbell, S. Lau, and T. Ligocki. Parallelization of Radiance For Real Time Interactive Lighting Visualization Walkthroughs. In *Proceedings of Supercomputing Conference (SC'99)*, Portland, OR, November 1999.

[68] L. Savioja, J. Huopaniemi, T. Huotilainen, and T. Takala. Real-Time Virtual Audio Reality. In *Proceedings of ICMC 1996*, pages 107–110, August 1996.

[69] J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. L. Hennessy. Load Balancing and Data Locality in Adaptive Hierarchical N-body Methods: Barnes-Hut, Fast Multipole and Radiosity. *Journal of Parallel and Distributed Computing*, 27(2):118–141, June 1993.

[70] N. Tsingos and J. D. Gascuel. Fast Rendering of Sound Occlusion and Diffraction Effects for Virtual Acoustic Environments. In *104-th AES Convention*, pages 4–7, Amsterdam, The Netherlands, May 1997.

[71] A. Verstak, J. He, L. T. Watson, T. S. Rappaport, C. R. Anderson, K. Bae, J. Jiang, and W. H. Tranter. S^{4}W: Globally Optimized Design of Wireless Communication Systems. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)*, Fort Lauderdale, Florida, April 2002.

[72] I. Wald and P. Slusallek. State of the Art in Interactive Ray Tracing. In *Proceedings of Eurographics*, pages 21–42, Manchester, United Kingdom, September 2001.

[73] J. Walfisch and H.L. Bertoni. A Theoretical Model of UHF Propagation in Urban Environments. *IEEE Transactions on Antennas and Propagation*, 36(12):1788–1796, December 1988.

[74] M. S. Warren and J. K. Salmon. A Portable Parallel Particle Program. *Computer Physics Communications*, 87:1223–1235, 1995.

[75] B. Wilkinson and M. Allen. *Parallel Programming – Techniques and Applications using Networked Workstations and Parallel Computers*. Prentice Hall, Englewood Cliffs, NJ, 1997.

[76] G. V. Wilson. *Practical Parallel Programming*. MIT Press, Cambridge, MA, 1995.

[77] X. Zhao, J. Kivinen, P. Vainikainen, and K. Skog. Propagation Characteristics for Wideband Outdoor Mobile Communications at 5.3 GHz. *IEEE Journal on Selected Areas in Communications*, 20(3):507–514, April 2002.

A Load-Balancing Scheduling Schemes

We consider three scheduling schemes used by the manager to assign tasks to workers, the fixed-size-task, the variable-size-task, and the hybrid-size-task assignment schemes. Suppose that the number of computation units for the original problem is T, the number of remaining computation units is T_{rem}, the number of nodes in the NOW configuration is N, the size of each assignment is G computation units in the fixed-size-task scheme and the hybrid-size-task scheme, the adjustment factor is F ($0 < F < 1$) when the variable-size-task scheme or the hybrid-size-task scheme is used. We define A_f, A_v, and A_h as the sizes of the current assignment by these three schemes, respectively. We also define N_f, N_v, and N_h as the number of assignments by these three schemes. Then ($\lceil x \rceil$ is the smallest integer larger than x), $A_f = \min(G, T_{rem})$, $A_v = \lceil T_{rem}F/N \rceil$, and $A_h = \min\{\max(\lceil T_{rem}F/N \rceil, G), T_{rem}\}$. It is easy to show that $N_f = \lceil T/G \rceil$. To compute N_v, we let T_i be the number of remaining tasks at the i-th assignment and A_i be the size of the i-th assignment, then, $T_1 = T$ and $A_i = \lceil FT_i/N \rceil$. For simplicity, we drop the constraint of $\lceil \rceil$ in the following derivation. After the i-th assignment, the number of remaining tasks T_{i+1} becomes $T_{i+1} = T_i - A_i = (1 - F/N)^i T$. Suppose that at the k-th step, $A_k \leq 1$, but $A_i > 1$ for all $i < k$, then

$$A_k = \frac{FT_k}{N} \leq 1; T_k \leq \frac{N}{F}; (1 - \frac{F}{N})^{k-1} T \leq \frac{N}{F}; k \geq \frac{\log \frac{N}{FT}}{\log(1 - \frac{F}{N})} + 1$$

The remaining tasks are T_k and are assigned to the workers by using size of 1. So, the total number of assignments N_v is

$$N_v = (k-1) + T_k \geq \frac{\log(\frac{N}{FT})}{\log(1-\frac{F}{N})} + T_k \approx \frac{\log(\frac{N}{FT})}{\log(1-\frac{F}{N})} + \frac{N}{F}$$

Similarly, to calculate N_h, we let T_i be the number of remaining tasks at the i-th assignment and A_i be the size of the i-th assignment, then $A_i = \min\left\{\max\left(\lceil\frac{FT_i}{N}\rceil, G\right), T_i\right\}$. Again, for simplicity, the constraint of $\lceil\rceil$ is dropped in the following calculation. After the i-th assignment, the number of remaining tasks T_{i+1} becomes

$$T_{i+1} = T_i - A_i = T_i - \frac{FT_i}{N} = (1-\frac{F}{N})T_i = (1-\frac{F}{N})^2 T_{i-1} = \ldots = (1-\frac{F}{N})^i T_1 = (1-\frac{F}{N})^i T$$

Suppose that at the k-th step, the size of the assigned tasks is $A_k \leq G$, but $A_i > G$ for all $i < k$, then

$$A_k = \frac{FT_k}{N} \leq G; T_k \leq \frac{GN}{F}; (1-\frac{F}{N})^{k-1}T \leq \frac{GN}{F}; k \geq \frac{\log(\frac{GN}{FT})}{\log(1-\frac{F}{N})} + 1$$

For the remaining tasks (T_k), the fixed-size-task scheme with G is used. Therefore, the total number of assignments N_h is

$$N_h = (k-1) + \frac{T_k}{G} \geq \frac{\log(\frac{GN}{FT})}{\log(1-\frac{F}{N})} + \frac{T_k}{G} \approx \frac{\log(\frac{GN}{FT})}{\log(1-\frac{F}{N})} + \frac{N}{F} = N_v + \frac{\log G}{log(1-\frac{F}{N})}$$

Under the condition

$$\frac{N}{FT} \geq (1-\frac{F}{N})^{(\frac{T}{G}-\frac{N}{F})}$$

It is easily shown that $N_v \leq N_f$. Therefore, the performance of the variable-size-task scheme is better than that of the fixed-size-task scheme in terms of assignment rounds, while the hybrid-size-task scheme performs better than the variable-size-task scheme as long as $G \geq 1$.

B Computation-Duplication and Computation-Partition

Suppose that the lookup table to be established is $M = (m_{i,j})_{n\times n}$, $m_{i,j}$ is a double-precision floating point number ($i,j = 1,2,\ldots,n$), N is the number of nodes in NOW, the cost of broadcasting each $m_{i,j}$ is t_b (in seconds), the cost of computing each $m_{i,j}$ is t_p (in seconds). Then, for the Sequential method, we further let T_{seq} and W_{seq} be the processing time and the total workload when the computation is carried out by only one node, it is easy to show that, $T_{seq} = t_p n^2$ and $W_{seq} = t_p n^2$. For the computation-duplication method, we let T_{dup}, W_{dup}, S_{dup} and R_{dup} be the finish time, the total workload, speedup and the workload expansion ratio. Since each node carries out the same computation independently and proceeds simultaneously, then $T_{dup} = t_p n^2$, $W_{dup} = N t_p n^2$, $S_{dup} = T_{seq}/T_{dup} = t_p n^2/(t_p n^2) = 1$, and $R_{dup} = W_{dup}/W_{seq} = (N t_p n^2)/(t_p n^2) = N$.

For the computation-partition method, we let T_{par}, W_{par}, S_{par} and R_{par} be the finish time, total workload, speedup and workload expansion ratio. Also let T_p and T_c be the computation time and time spent on communication, since the computation of the lookup table is distributed uniformly among all nodes, each node processes n^2/N elements of M, then broadcasts the results to other nodes. Therefore, $T_p = t_p n^2/N$, $T_c = t_b(n^2/N)N = t_b n^2$, $T_{par} = T_p + T_c = t_p n^2/N + t_b n^2$, $W_{par} = NT_p + T_c = N t_p n^2/N + t_b n^2 = (t_p + t_b)n^2$, $S_{par} = T_{seq}/T_{par} = 1/N + t_b/t_p$, and $R_{par} = W_{par}/W_{seq} = 1 + t_b/t_p$.

The difference between T_{par} and T_{dup} is $T_{par} - T_{dup} = n^2\left[t_p(1/N - 1) + t_b\right]$. It is clear that $T_{par} < T_{dup}$ when $t_p > t_b$ and $N > t_p/(t_p - t_b)$. $T_{par} \geq T_{dup}$ otherwise.

C Intermediate Results Assembly

We consider two intermediate result assembling methods here, the first-come-first-serve (FCFS) method and multi-level assembly method. In FCFS assembly method, every worker sends its intermediate results to the manager, the manager collects and merges the intermediate results according to the policy of ”first come first serve”. While in the multi-level assembly method, a binary tree is formed among all sites. The assembly process begins at the leaf level where every site is a leaf node, Nodes are grouped pairwise and assembly proceeds simultaneously among all node-pairs. The assembled data within each pair is stored at only one node, and the latter will take part in the next round of assembly, while the other in the same pair will be idle from then on. This procedure is repeated until all data are assembled at one site (i.e., the manager site).

We assume that the number of sites in the NOW configuration is N, the size of intermediate data held by each site before assembly is the same and is denoted as L (in bits), all intermediate data are distinct (i.e., no overlap), the cost of sending one bit through the network is t_c (in seconds), the cost of processing one bit data is t_p (in seconds), T_{fcfs} and T_{ml} are the processing time for the FCFS method and multi-level assembly method, then, T_{fcfs} and T_{ml} can be calculated as follows. For the FCFS assembly method, we let T_c, T_p be the communication and data-processing overheads, respectively, then

$$\begin{aligned} T_c &= t_c L(N-1); \qquad T_p = t_p[2L + 3L + 4L + ... + NL] = \frac{1}{2}(N^2 + N - 2)Lt_p \\ T_{fcfs} &= T_c + T_p = t_c L(N-1) + \frac{1}{2}(N^2 + N - 2)Lt_p \end{aligned}$$

For the multi-level assembly method, we tag the binary tree with level identifier by denoting the leaf level as level 1, and the root is at level $\log_2(N)$. Let c_i and d_i be the total communication time and data-processing time at level i, and l_i be the size of the assembled data, then

$$\begin{aligned} & c_1 = t_c L; \qquad d_1 = t_p(2L); \qquad l_1 = 2L; \\ & c_2 = t_c(2L); \qquad d_2 = t_p(2^2 L); \qquad l_2 = 2^2 L; \\ & c_3 = t_c(2^2 L); \qquad d_3 = t_p(2^3 L); \qquad l_3 = 2^3 L; \\ & \ldots \\ & c_{\log_2(N)} = t_c(2^{\log_2(N)-1} L); \qquad d_{\log_2(N)} = t_p(2^{\log_2(N)} L); \qquad l_{\log_2(N)} = 2^{\log_2(N)} L = NL \\ & T_{ml} = \sum_{i=1}^{\log_2(N)} (c_i + d_i) = \sum_{i=1}^{\log_2(N)} \left[t_c(2^{i-1} L) + t_p(2^i L) \right] \\ & = t_c L(N-1) + 2t_p L(N-1) \end{aligned}$$

The difference between T_{ml} and T_{fcfs} is $T_{ml} - T_{fcfs} = -t_p L(N-1)(N-2)/2$. It is clear that as long as $N \geq 2$, we then have $T_{ml} \leq T_{fcfs}$. Therefore, the multi-way assembly method is always better than FCFS assembly method.

D Prediction Results Generation

We consider two methods to generate predictions by collecting and assembling information scattering among all sites, the “one-site-pruning” method and the “all-site-pruning” method. In the first method, the removal of all non-significant raypaths for receivers is performed by the manager only. While in the second method, all sites take part in the elimination of non-significant raypaths, generate partial predictions, and the final predictions are assembled by the manager. The 4-step procedure for the “all-site-pruning” method by using collective communication functions is:

1. Exchange maximum powers for receivers. All sites compute the maximum power for each receiver based on their local information, then exchange this information among themselves, and find out the global maximum power for each receiver by using the global reduction operation with operation of `MPI_MAX` in `MPI_Allreduce(·)` function. The reduction results are returned to all sites.
2. Eliminate non-significant raypaths. Each site removes all non-significant raypaths for each receiver based on the global maximum power generated at the previous step.
3. Generate partial predictions. Each worker computes partial predictions based on its local information.
4. Generate the final predictions. By using another collective communication function, `MPI_Reduce(·)` with operator of `MPI_SUM`, all sites deliver their partial predictions to the manager and the final predictions are formed at the manager's site.

To calculate the processing time for these two methods, we assume that the number of sites in the NOW configuration is N, the number of receivers is N_{rec}, at each site, each receiver is illuminated by N_{ray} raypaths, each raypath can be described by B_{ray} (in bits), f_{ray} of N_{ray} are significant raypaths, the cost of sending one bit through the network is t_c (in seconds), the cost of processing one bit of input data or generating one bit data for predictions is t_p (in seconds), the cost of processing one bit data in the global reduction operation is t_g (in seconds), the power strength and the prediction parameter can be described by a double-precision floating-point number and has size of B_p (in bits), and T_{one} and T_{all} are the processing time for the "one-site-pruning" method and "all-site-pruning" method, respectively, then, for the "one-site-pruning" method, by denoting T_c, T_r, and T_p as the costs for communication, non-significant raypath removal, and predictions generation, respectively, we have

$$T_c = (N-1)t_c N_{rec} N_{ray} B_{ray}; \qquad T_r = N t_p N_{rec} N_{ray} B_{ray}; \qquad T_p = N t_p f_{ray} N_{rec} N_{ray} B_{ray}$$
$$T_{one} = T_c + T_r + T_p = \left[(N-1)t_c + N(1+f_{ray})t_p\right] N_{rec} N_{ray} B_{ray}$$

While for the "all-site-pruning" method, by denoting T_i as the processing time for the ith step (i = 1, 2, 3, 4) in the 4-step procedure for the "all-site-pruning" method presented above, we have

$$\begin{aligned}
T_1 &= t_g N_{rec} B_p; \qquad T_2 = t_p N_{rec} N_{ray} B_{ray}; \\
T_3 &= t_p f_{ray} N_{rec} N_{ray} B_{ray}; \qquad T_4 = t_g B_p; \\
T_{all} &= \sum_{i=1}^{4} T_i = t_g N_{rec} B_p + t_p N_{rec} N_{ray} B_{ray} + t_p f_{ray} N_{rec} N_{ray} B_{ray} + t_g B_p \\
&= t_g (N_{rec}+1) B_p + (1+f_{ray}) t_p N_{rec} N_{ray} B_{ray}
\end{aligned}$$

The difference between T_{all} and T_{one} is

$$\begin{aligned}
T_{all} - T_{one} &= t_g (N_{rec}+1) B_p + (1+f_{ray}) t_p N_{rec} N_{ray} B_{ray} \\
&\quad - \left[(N-1)t_c + N(1+f_{ray})t_p\right] N_{rec} N_{ray} B_{ray} \\
&= t_g (N_{rec}+1) B_p \left\{ 1 - \frac{(N-1)\left[t_c + (1+f_{ray})t_p\right]}{t_g} \frac{N_{rec} N_{ray} B_{ray}}{(N_{rec}+1) B_p} \right\}
\end{aligned}$$

It is very easy to achieve $T_{all} \le T_{one}$ in (1) as long as the network bandwidth is high and N is relatively large.