On the Parallel Simulation of Scale-Free Networks

Robert Pienta and Richard M. Fujimoto
School of Computational Science & Engineering
Georgia Institute of Technology
pientars@gatech.edu,
fujimoto@cc.gatech.edu

ABSTRACT

Scale-free networks have received much attention in recent years due to their prevalence in many important applications such as social networks, biological systems, and the Internet. We consider the use of conservative parallel discrete event simulation techniques in network simulation applications involving scale-free networks. An analytical model is developed to study the parallelism available in simulations using a conservative time window synchronization algorithm. The performance of scale-free network simulations using two variants of the Chandy/Misra/Bryant synchronization algorithm are evaluated. These results demonstrate the importance of topology in the performance of synchronization protocols when developing parallel discrete event simulations involving scale-free networks, and highlight important challenges such as performance bottlenecks that must be addressed to achieve efficient parallel execution. These results suggest that new approaches to parallel simulation of scale-free networks may offer significant benefit.

Categories and Subject Descriptors

I.6 [**Computing Methodologies**]: Simulation and Modeling

Keywords

conservative synchronization; discrete event simulation; parallelism; power law degree distribution; scale-free; scale-free degree distribution; scale-free network simulation; simulation synchronization

1. INTRODUCTION

Parallel discrete event simulation (PDES) offers a means to accelerate, and/or enable large-scale simulations by distributing the execution of a simulation program over multiple processors. It has been applied to a wide variety of applications such as simulating telecommunication networks [24], surface and air transportation systems [29, 31], and disease spread [23], to mention a few. A PDES program consists of a collection of logical processes (LPs) that communicate by exchanging timestamped messages, or events.

A synchronization mechanism is needed to ensure the parallel execution of the simulation produces exactly the same results as the corresponding sequential execution. Synchronization algorithms are broadly classified as conservative mechanisms that rely on blocking logical processes in order to ensure events within each LP are always processed in non-decreasing timestamp order, and optimistic algorithms that allow out-of-order event processing to occur, but utilize a rollback mechanism to recover when events have not been processed in timestamp order. A critical factor impacting parallel performance is the amount of overhead, e.g., additional messages or rolled back computation, that must be introduced by the synchronization mechanism. Managing this overhead is critical to achieving efficient executions.

The PDES topology is a graph with each node representing an LP, and each link connecting two LPs indicating those LPs may communicate by sending a message between them. Here, we assume undirected arcs, and do not distinguish between the sending and receiving LP except to clarify the presentation.

We are concerned with one important class of network topologies that have become known as *scale-free networks* [4]. A scale-free network is one where the node degree follows a power law distribution. A distinguishing characteristic of scale-free networks is a significant number of nodes referred to as *hub* nodes contain a large node degree, while most nodes, often referred to as *leaf nodes*, contain relatively small degree. Scale-free networks are known to exhibit "small world" properties, meaning, the minimum path length between any pair of nodes in the network is small relative to the number of nodes in the network. As will be seen later, these properties have important ramifications for parallel discrete event simulation.

Scale-free networks have received a considerable amount of attention in recent years because it has been observed that many real-world systems contain networks that exhibit the scale-free property [30]. For example, it is widely believed that the autonomous system (AS) level topology of the Internet is scale-free [11, 25, 13, 33]. In the broad area of systems biology, the study of complex biological systems, protein-protein interaction networks have been demonstrated to follow scale-free distributions [16]. Some financial networks such as the interbank payment network exhibit scale-free behavior [27] Social networks, the world-wide-web, the internal structuring of superconductors, the airline transporta-

SIGSIM-PADS'13, May 19–22, 2013, Montréal, Québec, Canada.

tion network, and human interaction networks such as those characterizing the spread of diseases have been observed to exhibit the scale-free property.

Other networks, while not following a power-law distribution, do appear to have heavy skew in their node degree distribution. For example, initial studies suggested that the router-level topology of the Internet followed a power law, but more recent work reports the topology is driven by engineering and economic considerations as well as physical node and link capacity constraints, resulting in a node degree distribution that is skewed, but does not follow a power law [32]. The router-level topology contains many high-degree routers at the edge of the network designed to aggregate traffic from many low-bandwidth connections coupled with a mesh-like core of high capacity routers with a smaller number of high bandwidth links. Although the work described here focuses on networks with a power law degree distribution, we conjecture that many of the results derived in this study also have applicability to networks with heavily skewed degree distribution.

We are concerned with the efficiency of parallel simulation techniques in terms of parallelism and overhead of synchronization protocols for scale-free networks. It seems clear that topology will have strong implications regarding performance. For example, it has been observed empirically that the distribution of event-level parallelism in simulations of communication networks can lead to severe load imbalances [19]. In [9] load distribution issues in scale-free network simulations are examined. However, to our knowledge, the relationship between power law topology and parallel simulator performance has not been previously studied at a foundational level.

The remainder of this paper is organized as follows. The next section describes qualitatively the relationship between scale-free network topologies and overhead of PDES synchronization protocols. While an exhaustive examination of all synchronization protocols is beyond the scope of this work, this section discusses examples from major classes of synchronization protocols. Section 3 reviews the power law degree distribution that characterizes scale-free networks. Section 4 develops an analytical model to assess the parallelism available in parallel simulations of scale-free networks using a simple, synchronous window-based PDES protocol. This is followed by results from simulation studies that consider parallelism and overhead using two variations of the well-known Chandy/Misra/Bryant protocol. Finally we present conclusions and areas of future research.

2. PDES SYNCHRONIZATION

Most PDES synchronization protocols were designed to be used to simulate networks of arbitrary topology. One exception is the protocol described in [17], that was designed for feedforward networks. Conservative synchronization algorithms broadly fall into two general categories. Synchronous algorithms utilize global synchronization points, e.g., barriers, to synchronize the computation. By contrast, asynchronous algorithms use only local synchronization mechanisms. One well-known example of the latter is the Chandy-Misra-Bryant (CMB) [6] algorithm.

Briefly, CMB sends a *null message* to another LP to indicate a lower bound on the timestamp of any message it might send in the future. The receiver uses this information to determine a lower bound on the time stamp (LBTS) of all future messages it might receive. Events with timestamp smaller than this LBTS value may be safely processed. In its simplest form, each LP broadcasts a null message on each of its outgoing links after processing each event. We refer to this approach as naïve CMB.

The number of links connected to an LP, i.e., the node degree, impacts the performance of the algorithm in two ways. First, an updated LBTS must be computed by each LP. In the worst case the number of comparisons is $O(logN_L)$, where N_L is the node degree of LP L. This calculation takes little time, but must be repeated often, as much as once for each received message. More importantly, however, an LP must send a null message to each of its neighbors, as often as after each message that it processes. This overhead is clearly proportional to node degree. This overhead is problematic for scale-free networks because high degree nodes will incur a significant overhead to send, receive, and process null messages. As will be seen later, hub nodes tend to have a disproportionately large amount of event computation that can cause them to become bottlenecks. The aforementioned overheads clearly aggravate this situation.

An alternative version of CMB uses a demand-driven approach [28]. Rather than broadcasting a null message at each message send, an LP L requests a null message from each LP when it must block to wait on an empty message queue. This will significantly reduce the number of null messages that are required. However, even with this optimization, one would expect the hub nodes to have to request more null messages than leaf nodes due to the large number of incoming links. Later, we use simulations to evaluate this question further.

Synchronous algorithms typically repeat the following steps: (1) compute LBTS values to determine which events are safe to process, (2) process those events with timestamp less than the LP's LBTS value and send any messages produced in processing these messages, (3) perform a barrier synchronization. Each iteration through these steps is referred to as an *epoch*. One of the simplest algorithms is YAWNS [22] that completes the first step by computing a global minimum that specifies the smallest timestamped message that could be generated, and uses this value as the LBTS. The performance of the YAWNS protocol for scale-free networks will be examined in much greater detail later.

Other synchronous algorithms put greater effort into computing the LBTS values. A common approach is to use the *distance between objects* [3] to determine the minimum amount of simulation time that must elapse for an event in one LP to affect an event in a different LP. To accomplish this, an LP must consider the smallest timestamp event within each other LP, and use the precomputed distance to compute an LP's LBTS value. This may require examining many other LPs to determine which events are safe. The bounded lag approach uses a simulation time window to limit the scope of this search, thereby arguing that the protocol scales to large numbers of processors [20]. The events within the window are considered *near future* events while those outside the window are considered *far future* events. The width of the window is important to ensure runtime efficiency; too small a window and the LP's won't be able to process very many events per epoch. Too large a window and the events that should be far future are included in the the calculations, again decreasing the runtime efficiency of the simulation.

Scale-free networks are a challenging test case for these synchronization algorithms. This is because scale-free networks possess the so-called *small world* property. Popularized in the media, small world networks have the property that the average minimum path length between two randomly selected nodes in the network tends to be very small, even for networks containing a large number of nodes. As discussed momentarily, power law distributio'ns are characterized by the parameter λ that determines the number of high degree nodes that will occur. In cases with $2 \leq \lambda \leq 3$, a common range for natural scale-free networks, the average path length grows even slower ($\log \log N$) [8]. Stated another way, the number of nodes (LPs) reachable from a given node increases exponentially with the number of hops. This suggests that algorithms that require examination of other LPs to determine what events are safe to process are doomed to failure because a very large number of nodes will have to be examined to determine what events are safe. Windowing schemes such as that used in bounded lag are unlikely to solve this problem because the relevant "neighborhood" of nodes that must be examined grows too rapidly.

Network topology may also impact the performance of optimistic synchronization algorithms [5]. Approaches such as the well-known Time Warp algorithm [15] utilize a rollback propagation mechanism that can severely degrade performance. Analysis of time warp systems has been carefully studied through a number of different approaches. Overall parallel performance bounds have been studied through modeling the critical path within simulations [18]. The general performance of time warp systems has also been rigorously investigated assuming a fully connected network topology [14, 1]. The propagation of rollbacks through the network is dependent on network topology; however, the relationship of the rollback propagation mechanism to topology is complex, and not well understood. Our focus here is to explore parallelism in scale-free network simulations and to elucidate the behavior of conservative synchronization approaches on scale-free topologies. Here, we focus exclusively on conservative synchronization algorithms.

3. SCALE-FREE NETWORKS

For the examination of scale-free networks it is generally assumed that the node degree probability is [7]:

$$P(k) \sim ck^{-\lambda}$$

Where k is the node degree, c is a scalar normalization constant, and λ is a parameter. Intuitively, smaller values of λ indicate the network will contain more nodes with large degree. Typical values of λ for networks that appear in practice generally fall in the range $\lambda \in (1, 3]$ [4, 7]. Early investigations of natural systems by Italian economist Vilfredo Pareto found that many observable phenomena follow a Pareto distribution [2].

The Pareto cumulative distribution function follows:

$$F(x) = \begin{cases} 1 - \left(\frac{m}{x}\right)^{\alpha} & \text{for } x \geq m, \\ 0 & \text{for } x < m \end{cases}$$

where m is the positive, minimum degree across all vertices in the network. The Pareto distribution is a reformation of the general power-law distribution shown above:

$$\left(\frac{m}{x}\right)^{\alpha} = m^{\alpha}\left(\frac{1}{x}\right)^{\alpha} = m^{\alpha}\,(x)^{-\alpha}$$

In this case $c = m$ and $\alpha = \lambda$. This leads to to the probability mass function:

$$F(x)' = f(x) = \begin{cases} \alpha\left(\frac{m^{\alpha}}{x^{\alpha+1}}\right) & \text{for } x \geq m, \\ 0 & \text{for } x < m \end{cases}$$

From this the expectation can be formulated:

$$E(X) = \int x f_X(x) dx = \frac{\alpha m}{\alpha - 1}, \text{ for } \alpha > 1$$

These definitions will be called upon later during the analysis of scale-free network behavior to yield quantitative information about the network.

4. CONSERVATIVE PDES ANALYSIS

4.1 Application Model and Execution

We have developed a general model of synchronous simulation using the YAWNs protocol described earlier. The system utilizes an event processing paradigm based on the well-known PHOLD model [12] adapted for scale-free networks. The system initially contains a fixed set of messages that are uniformly distributed across all LPs in the system. When an event is processed, exactly one new event is created and sent to a destination LP that is chosen at random from a uniform distribution among the LP's neighboring nodes. We assume each LP has a lookahead of L, and the timestamp increment for each newly created event is L plus a value drawn from an exponential distribution. We assume each LP is mapped to a separate processor. It is assumed that each event requires one unit of wall clock time to process that event.

In the synchronous YAWNS protocol a fixed sized window is used whose size is equal to the lookahead L of each LP. All events with timestamp in the window $[Min, Min + L)$ are processed in each epoch, where Min is the minimum timestamp event among those that have not yet been processed. We assume the time to compute LBTS values and communication overheads, e.g., to send and receive events, are negligible, and event computations are not preempted once the processing of an event begins. Therefore, the time to complete an epoch is proportional to the maximum number of events among the LPs whose timestamp reside within the current window.

4.2 Analytical Model

The main goal of the analytical model is to analyze the affects of the underlying topology on the amount of parallelism in the simulation. To aid this investigation we created a stochastic analytical model to estimate the steady state arrival of events within an epoch of width L. In window-based synchronous approaches, the LP with the longest runtime will become the bottleneck. When the number of events within each time window among the different LPs vary widely, the underutilized LPs will become idle, waiting for the LP containing the largest number of events. This results in a loss of parallelism and reduces the efficiency of the distributed simulation. In order to empirically measure the parallelism in the model, our approach calculates the available parallelism as follows: Let the number of events per LP_i for a single epoch be n_i. The bottleneck LP, or longest running LP in wallclock time, (LP-4 in fig. 1) is:

$$\arg\max\{n_i\}$$

Since the processing time of each event is 1 unit of wallclock time, the amount of time to complete the epoch is e_{max} (see fig. 1) where:

$$e_{max} = \max\{n_i\}$$

The total amount of work, i.e., the number of events within the epoch is W (represented as the events inside the shaded region) can then be calculated:

$$W = \sum_i n_i$$

The amount of parallelism, P, is:

$$P = \frac{W}{e_{max}}$$

In other words, P is the maximum speedup that can be obtained by a parallel execution of the simulation.

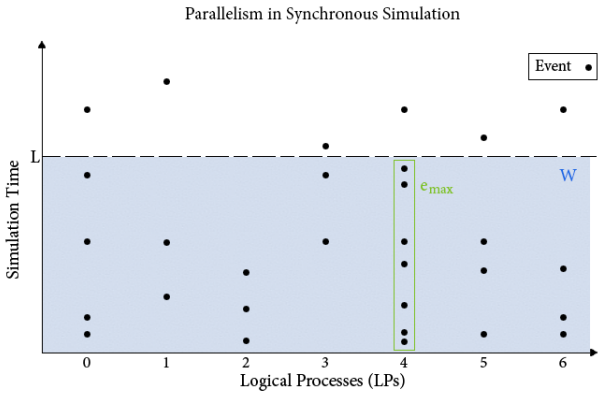

Figure 1: The measure of parallelism, P, is based on the LP with the largest number of events and the total amount of time spent processing events in the epoch. In this example the shaded region denotes a single epoch with 7 LPs. LP-4 determines the value of $e_{max} = 7$ and the total number of events $W = 24$ is the number of events in the shaded region.

The exact arrival time distribution for a message is complex, so we have adopted a number of approximations and simplifications for its form. Many of the approximations are common when attempting to analytically model a simulation system [10, 22]. Numerical sample estimates can then be computed for each of the parameters. This is predicated on the notion that the arrival time, t, of a particular event can be viewed as the sum of some number of past service events. This analysis is novel in that it considers the effect of a scale-free or power-law degree distribution within the simulation network topology. In the following analysis we taxonomize nodes in the simulation network as either "leaf" or "hub" nodes. Leaf nodes are nodes of relatively low degree while the hub nodes are those of high degree. Let the top f fraction of nodes be hubs while the remaining $1 - f$ fraction of nodes are leaves.

With the message arrival times following an approximate distributional form, it is possible to compute a new probability distribution of the number of events executed within a known period of simulation time. Similar to Dickens et al. the analysis presented in this work focuses on epochs or generations of messages across a known period of simulation time [10]. Each generation is a window of simulation time $[t, t + L)$. All events with timestamp $e_t \in [t, t + L)$ are considered to be members of that generation. As events are processed within the first generation some number of the new events will fall within the next or subsequent generations and are not counted among the events in the current generation.

The number of service events that an LP starts in $[t, t+L)$ is a random variable e_{serv} that follows a Poisson distribution with mean determined by the previous epoch. Because the routing of an event in a scale-free network is not uniform, the number of job arrival events is a random variable J, conditioned on the type of node, i.e., hub or leaf.

Given N messages at generation k the number arriving at an LP is a binomial distribution conditioned on the node type. Because the connectivity between nodes is by definition not uniform, this conditioning is used to model the average case hub and leaf nodes. Let u_i and v_i be the number of events at a hub and leaf at generation i respectively. The values V_h and V_l are the expected number of edges for each type of node. The resulting binomial for hubs is then $B(u_i, 1/V_h)$ and $B(v_i, 1/V_l)$ for leaves. Because the hubs are picked as some fraction, f, of the most highly connected nodes, there exists a well defined point x_f after which lies a fraction f of the measurements.

$$\int_{x_f}^{\infty} cx^{-\lambda}\,dx = f\int_{x_{min}}^{\infty} cx^{-\lambda}\,dx$$

Given $\lambda > 1$, x_f exists and can be solved for; yielding:

$$x_f = \left(\frac{1}{f}\right)^{1/\lambda-1} x_{min}.$$

With the boundary point x_f, it is possible to separately calculate the expected degree in the points up to x_f (leaves) and after it (hubs). In the case of hubs with $\lambda > 2$:

$$V_h = \int_{x_f}^{\infty} xcx^{-\lambda}\,dx = -\frac{c}{-\lambda+2}x_f^{-\lambda+2}$$

The expected the degree for the leaves is similar:

$$V_l = \int_{x_{min}}^{x_f} xcx^{-\lambda}\,dx = \frac{c}{-\lambda+2}\left[x_f^{-\lambda+2} - x_{min}^{-\lambda+2}\right]$$

For a power-law distributed network a similar derivation as the one above can be used to show the fraction of edges going to some fixed fraction of the most highly connected nodes. Consider a scale-free network where E denotes the fraction of the edges connected to some fraction f of the highest degree nodes, then the following equation relates E to f:

$$E = f^{\frac{\lambda-2}{\lambda-1}}$$

This solution, derived first in [21], yields an important measure of the probability that a randomly selected edge will lead to either a hub or a leaf, determined entirely by the parameters of the scale-free distribution. For example, if we take the top 50% of nodes to be the hubs on a scale-free network with $\lambda = 2.1$, we can say that $0.5^{\frac{0.1}{1.1}} \approx 0.938$ or that approximately 93% of all edges will go to hubs while only 7% will go to leaves. Considering the equation above, the number of hub edges per leaf edge increases heavily as

$\lambda \to 2$. The imbalance of edges may cause disproportionate amounts of communication with the hubs and therefore decrease the overall load balance.

The expected number of arrival events a leaf expects to see, R_l, is modeled by the expected number of edges for a leaf, split into fN hubs and $(1-f)N$ leaves for a network with N nodes, each scaled by the average case probability of receiving a message on that edge. We use f to partition the nodes into two sets for the analysis. With the approximate number of total edges, we must separate the expected number of edges from hubs and leaves and then scale by the probability that a message was sent down those channels. By calculating leaves and hubs separately, the total number of incoming messages can be calculated. First for a leaf:

$$Rl_i = V_l(1-f) \cdot E[B(v_{i-1}, 1/V_l)] + V_l f \cdot E[B(u_{i-1}, 1/V_h)]$$

and for a hub:

$$Rh_i = V_h(1-f) \cdot E[B(v_{i-1}, 1/V_l)] + V_h f \cdot E[B(u_{i-1}, 1/V_h)]$$

Often such binomial distributions are approximated with Poisson random variables of matched mean. Because $1/V_l$ may be relatively large, our approach cannot safely make this substitution. Here we make use of the expectation of a binomial $E[\, B(N,p)\,] = Np$ resulting in:

$$Rl_i = V_l(1-f)\frac{v_{i-1}}{V_l} + V_l f \frac{u_{i-1}}{V_h}$$

for the average leaf node and,

$$Rh_i = V_h(1-f)\frac{v_{i-1}}{V_l} + V_h f \frac{u_{i-1}}{V_h}$$

for the average hub node.

Let the service time of events be an exponential random variable, with rate μ_s. We are interested in determining the expected number of messages that are scheduled in the *ith* subsequent generation. If the current time is t, the *ith* subsequent generation is the period in simulation time for which the events timestamps are $[t+(i+1)L, t+(i+2)L)$.

$$P\{\textit{Service entry arrival time} \in \textit{ith generation}\}$$

$$= \int_{t+(i+1)L}^{t+(i+2)L} e^{-\mu_s u} du$$

$$= \frac{1}{\mu_s}\left(e^{-\mu_s(t+iL+L)} - e^{-\mu_s(t+iL+2L)}\right)$$

With the above probability, the mean number of events that fall outside of the first generation into each of the *ith* subsequent generations can be calculated. This value is added to the expeceted number of messages in that generation. This calculation must be done at each iteration to mimic the effects of scheduling future events.

Leaf:

$$E[Gl_0] = Gl_{init}$$

Hub:

$$E[Gh_0] = Gh_{init}$$

Initial values for the number of events in the first generation, Gh_{init} and Gl_{init} are required. The next generation is a recurrence based on the previous value, because the number of messages received at generation i dictates the number of messages to be dealt with in the next and subsequent generations. Given the first generation expected values, later generations within this model may be determined.

Leaf:

$$E[Gl_i] \mathrel{+}= Rl_i$$

Hub:

$$E[Gh_i] \mathrel{+}= Rh_i$$

The recurrence lies within the calculations of Rl_i and Rh_i and the number of events already scheduled in the current generation. Given initial conditions to determine the first generation, subsequent generations can be solved in an interative manner. This iterative analysis steps generation by generation over the course of a simulation. From these results the expected number of messages per epoch as well as some of the scale-free network behavior can be gleaned.

4.3 Largest Node Degree

In order to investigate the workload of the largest hub, an estimate of that node's degree is needed.

Given n samples drawn from a power-law distribution, we first compute the probability that the largest value, x_{max}, lies in the interval between x and $x+dx$. Let $P(x)$ be the probability that $x \geq X$, then:

$$P(x) = \int_x^\infty p(x')dx' = \left(\frac{x}{x_{min}}\right)^{-\lambda+1}$$

We are concerned with the total proportion of nodes with degree greater than or equal to x_{max}. Consider the following heuristic argument:

$$\frac{P(x \geq x_{max})}{P(x \geq x_{min})} \approx \frac{1}{n},$$

for some x, then there will be on average a single sample in the range x to ∞. This single sample will be the largest value. Given that

$$\frac{\int_{x_{max}}^\infty cx^{-\lambda}dx}{\int_{x_{min}}^\infty c'x^{-\lambda}dx} \approx \frac{1}{n},$$

solving for x_{max}:

$$x_{max} \approx x_{min} \cdot n^{\frac{1}{\lambda-1}}.$$

In the case where the minimum out-degree across all nodes is one ($x_{min} = 1$), the equation above simplifies further to

$$x_{max} \approx n^{\frac{1}{\lambda-1}}.$$

For a more rigorous derivation that arrives at the same approximation see the appendix of [21].

This result is important because the hub with the largest degree will likely form the bottleneck when using the synchronous simulation protocol. For $\lambda > 2$ the largest hub grows at a fractional power of the hubs. However when $\lambda = 2$ the largest degree grows linearly with the number of vertices and when $\lambda < 2$ the rate of growth is superlinear. For example let $\lambda = 1.8$ the maximum degree grows at a rate approximately $n^{1.25}$ [21, 26]. Although in a real network the maximum degree reaches a hard limit of the number of vertices in the graph, the value of λ determines how quickly x_{max} reaches this point.

4.4 Numerical Experiments and Results

For illustrative purposes, consider the analytical model introduced previously assuming the top 25% of nodes are hubs in a scale-free network with $\lambda = 2.5$. Let $x_{min} = 1$, such that all nodes have at least one edge connecting them in the network. Initially, each node has the same initial number of events. In the following experiments an initial set of 10 events per LP was used. As mentioned in the previous section, each event that is processed produces exactly one new event with timestamp increment drawn from an exponential distribution. The model is iterated until a steady-state is reached.

This iteration is repeated for leaf, hub, and the largest (highest degree) hub expected for a particular network size N. The steady state expected number of messages per node type is calculated, with an estimate of the number of messages the largest hub would receive. The minimum time for completion, e_{max}, as determined by the longest running LP is calculated. In all cases tested e_{max} was determined by the largest hub. The total number of messages processed W is calculated. From these values P emerges. This process is repeated for scale-free distributions with varied λ's.

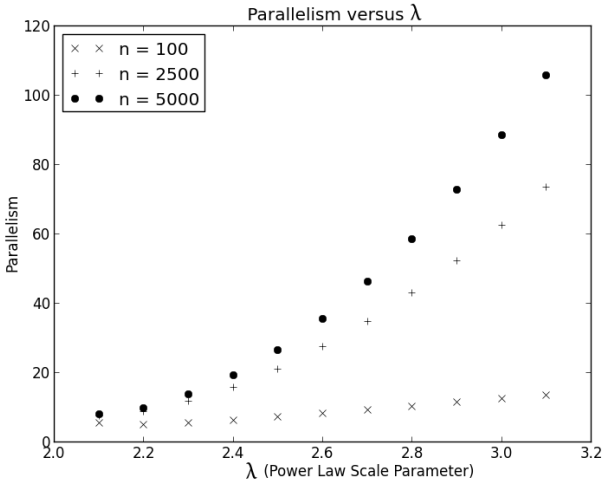

Figure 2: The model's potential parallelism given varied λ and network size.

Consider the behavior shown in figure 2. As expected, the potential parallelism increases with the number of nodes, however it is seen to be strongly dependent on λ. Because the derivation of the analytical model requires that $\lambda > 2$, the experiments focused on values larger than 2. While the analytical model cannot be used to for networks where $\lambda < 2$, the behavior of those scale-free networks will be more imbalanced than the networks tested here. This arises from the fact that as $\lambda \to 1$ the distribution becomes more tail-heavy, with the bulk of edges being pushed toward the hubs.

Given the analytical synchronous model above, one is naturally interested in the relevant λ's for networks that arise in practice. We have calculated the available parallelism for a telecommunications dataset. The dataset represents the collection of phone calls made during a single day among a large group of entities who made or received at least one call. The calculation of P utilizes several summary parameters from the dataset; network size (number of entities), minimum number of connections or calls (one call) and approximations of empirical λ for the dataset. For $\lambda = 2.22$ [21] and a network containing 1.1 million nodes the empirical data yielded a potential parallelism of only $P = 55.21$. As an approximate upper bound of the potential parallel improvement for simulating this network in a distributed setting, this value suggests that the scale-free nature of the network topology will significantly limit the amount of speedup that can be obtained using parallel simulation. The parallelism was limited largely by the low λ value which was caused by a few hub-entities making a large numbers of calls; likely these originated from businesses or telemarketing groups.

4.5 Discussion

The analytical model for synchronous approaches makes some assumptions that increase it's sensitivity to changes in λ. For example, the system utilizes the aforementioned relation, $E = f^{\frac{\lambda-2}{\lambda-1}}$, to calculate the proportion of edges connected to hubs. This relation is derived from a set of integrals that diverge when $\lambda \leq 2$. This does not necessarily mean that the averages themselves will all be infinite, but rather that individual measurements may be so large and varied that the average never converges. What happens analytically as $\lambda \to 1$ is that the tail of the distribution becomes heavier and heavier. In terms of a network topology this means that the number of edges going to the hubs increases until the point of saturation, wherein every edge is connected to a node in the set of hubs.

As expected, parallelism increases with the number of LPs (see fig. 4). However the amount of parallelism increases at only a modest rate. This is because the hub nodes become a bottleneck, limiting parallelism increases as more LPs are added. The parallelism increases more slowly for lower values of λ because these networks have more heavily loaded hub nodes. The higher the degree of the hubs nodes, the more events they receive and the longer it takes to process these events in a given epoch. Because the calculation of parallelism is determined by the longest epoch runtime (realized by the highest degree hub), there is less parallelism available in these networks. This suggests that the λ value of a scale-free network has significant implications on the amount of speedup that can be obtained.

5. SIMULATION RESULTS

We now describe the results of simulation experiments using the naïve and demand-driven CMB protocol to evaluate performance and overhead in simulating scale-free networks. A graph with an approximate power-law degree distribution of fixed λ was created via the Barabasi-Albert method implemented using a stochastic algorithm [4]. The experiments covered in this section were run on various sizes of scale-free networks. The simulated CMB algorithm begins by queuing simulated events with timestamp drawn from an exponential distribution. We assume that when each event is processed, a single new event is scheduled with timestamp increment drawn from an exponential distribution. We assume a finite, nonzero lookahead. A neighboring LP is then selected from a uniform distribution and the event is sent to that LP. This approach maintains a constant number of messages. A steady state behavior can be observed after an initial transient period. The number of null messages are counted as a measure of simulation overhead.

5.1 CMB Results

We calculate the parallelism for the CMB methods. Let e_{sim} be the number of simulation-relevant events–events that are not null messages–and e_{null} be the total number of null messages. Assume that each message, null or simulation-relevant, requires only 1 unit of wallclock time. The parallelism of the entire simulations can then be calculated as:

$$P = \frac{e_{sim}}{e_{sim} + e_{null}}$$

The following data were collected with experimental $\lambda = 2.5$. The top 25% of the most connected nodes were considered as the hubs. The mean null and event message counts were taken once the system reached steady state.

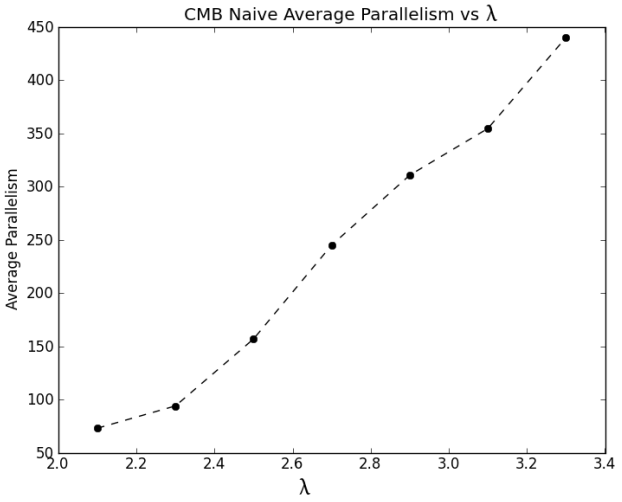

Figure 3: The average parallelism for the naïve CMB algorithm versus λ.

Experimental results for naïve CMB are presented in figure 3. The average parallelism as λ is varied is shown, and indicates that parallelism with asynchronous approaches is also driven by λ. The average parallelism for the naïve CMB algorithm increases as the value of λ increases. This agrees with the intuition that low values of λ cause a large amount of null message traffic to and from hub nodes.

The same experimental experimental conditions were used in the demand driven approach to assure experimental consistency. Even the parallelism of the CMB algorithm with the previously discussed optimizations is still susceptible to changes in λ; see figure 4. The demand driven CMB exhibited similar results to its naïve predecessor, albeit with significantly less overhead. Because this optimization reduces the amount of null message passing, it reduces the strain on the hubs when broadcasting null messages considerably; allowing for more parallelism than the naïve approach.

Figure 5 illustrates the number of null messages received by each distinct set of nodes. The hubs receive a constant factor more null messages than the leaves as the network size increases. Both asynchronous synchronization algorithms exhibited this behavior; again the demand driven CMB used fewer total null messages.

The mapping of LPs to processors in order to balance the workload is an important issue in any distributed simulation program. Given the rich get richer nature of heavy

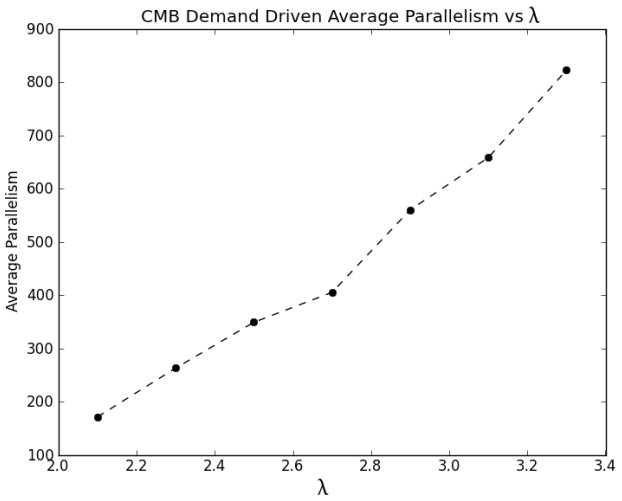

Figure 4: The average parallelism for the demand driven CMB algorithm versus λ.

tailed degree distributions, the amount of work and number of messages going to hubs can be wildly disproportionate.

5.2 Discussion

Both CMB parallelism experiments resulted in an increase in average parallelism as the value of λ increases. This is consistent with the observation that the heavy tails of scale-free networks lead to limited parallelism.

Both CMB experiments show an increasing number of overhead messages as the network grows. For the naïve case, we expect that the growth in null messages will scale superlinearly with the number of nodes. This growth arises from the naïve choice to broadcast the null messages to each neighbor. See figure 5.

This highlights potential issues regarding the scalability of these synchronization protocols for large scale-free networks. These experiments indicate that the difference in the load received and processed by LPs is a constant factor scaled by the size of the network and the particular λ value. The hubs become increasingly more severe bottlenecks as the size of the topology is increased. As both CMB experiments require a stochastically generated network sampled from a power-law degree distribution, they are less reliant on the edge-wealth relation and therefore more resilient to perturbations in λ.

6. CONCLUSIONS AND FUTURE WORK

Both the analytical model and simulation experiments highlight the highly unbalanced nature of parallel simulations of scale-free networks. The analytical model shows that hub nodes quickly become severe bottlenecks in parallel simulations. The exponent of the power-law distribution, λ plays an important role in determining the severity of the resulting hub node bottleneck. As λ decreases toward 1, the tail of the degree distribution becomes heavier, increasing the magnitude of the bottleneck. This results in the degree of the largest hub growing extremely quickly as network size is increased, especially when λ is near 2 or smaller. Our analytic model as well as the empirical simulation results

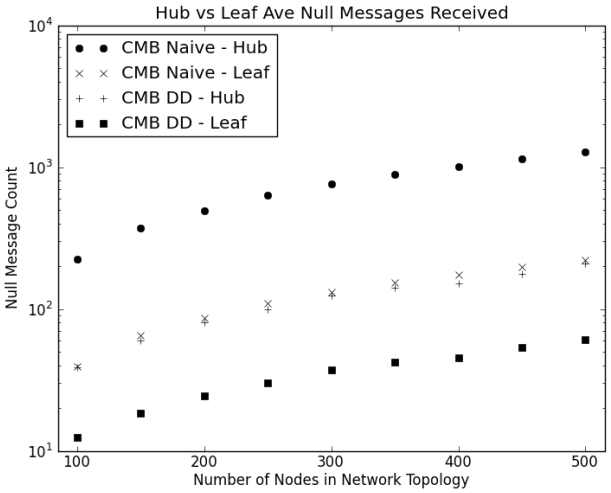

Figure 5: The overhead for both CMB approaches broken down by hub and leaf nodes.

showed that the value of λ has a strong influence on parallelism, and the potential speedup that may be obtained.

Networks with power law degree distribution appear in many natural and human-made systems. These studies illustrate that achieving efficient parallel execution of simulations of scale-free networks is a challenging task. There are many avenues of future research. The analytic models described here require further refinement and generalization for different classes of workload models. There are two important extensions to the analytical model; the first in the distribution of events across the LP's and the second to investigate the potential costs of LP-LP link delays. The impact of non-uniform event distributions across the LPs have not been deeply investigated and will help generalize the results to more realistic systems. The second extension to the model will allow us to model realistic synchronous distributed simulations with higher granularity by controlling the link delay distribution.

Models of other synchronization protocols are needed, e.g., other conservative protocols and optimistic synchronization mechanisms. More extensive experiments with real-world applications are needed to verify the observations reported in this paper appear in practice. Because scale-free and approximately power-law distributions appear in many important natural systems, a key facet of the future work will be spent examining distributed simulations running scale-free models.

Finally, many challenges exist to achieve efficient parallel simulation of large, scale-free networks. Additional research is required to identify effective synchronization methods for scale-free network simulations. Effective means to addressing the bottleneck issues of scale-free networks are needed. Scale-free networks have the potential to reveal interesting behaviors in complex systems. More real world data are needed relating the performance of new systems to their underlying topology.

7. REFERENCES

[1] I. F. Akyildiz, L. Chen, S. R. Das, R. M. Fujimoto, and R. Serfozo. The effect of memory capacity on time warp performance. *International Journal of Parallel and Distributed Computing*, 18(4):411–422, August 1993.

[2] B. C. Arnold. Pareto and generalized pareto distributions. 5:119–145, 2008.

[3] R. Ayani. *A Parallel Simulation Scheme Based on Distances Between Objects*. Technical report (Kungl. Tekniska högskolan. Dept. of Telecommunication Systems-Computer Systems). Royal Institute of Technology, Department of Telecommunication Systems-Computer Systems, 1988.

[4] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286:509–512, Oct. 1999.

[5] D. Bauer, G. Yaun, C. Carothers, M. Yuksel, and S. Kalyanaraman. Ross.net: optimistic parallel simulation framework for large-scale internet models. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 703 – 711 Vol.1, dec. 2003.

[6] K. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *Software Engineering, IEEE Transactions on*, (5):440 – 452, sept. 1979.

[7] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, Nov. 2009.

[8] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90, 2003.

[9] G. D'Angelo and S. Ferretti. Simulation of scale-free networks. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 20:1–20:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[10] P. M. Dickens, D. M. Nicol, P. F. Reynolds, Jr., and J. M. Duva. Analysis of bounded time warp and comparison with yawns. *ACM Trans. Model. Comput. Simul.*, 6(4):297–320, Oct. 1996.

[11] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, Aug. 1999.

[12] R. M. Fujimoto. Parallel discrete event simulation. *Commun. ACM*, 33(10):30–53, Oct. 1990.

[13] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery, 2000.

[14] A. Gupta, I. F. Akyildiz, and R. M. Fujimoto. Performance analysis of time warp with multiple homogeneous processors. *IEEE Trans. Softw. Eng.*, 17(10):1013–1027, Oct. 1991.

[15] D. Jefferson and H. Sowizral. Fast concurrent simulation using the time warp mechanism. 1982.

[16] H. Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, 2002.

[17] D. Kumar. Simulating feedforward systems using a network of processors. In *Proceedings of the 19th annual symposium on Simulation*, ANSS '86, pages 127–144, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press.

[18] Y.-B. Lin. Parallelism analyzers for parallel discrete event simulation. *ACM Trans. Model. Comput. Simul.*, 2(3):239–264, July 1992.

[19] X. Liu and A. A. Chien. Realistic large-scale online network simulation, 2004.

[20] B. D. Lubachevsky. Efficient distributed event-driven simulations of multiple-loop networks. *Commun. ACM*, 32(1):111–123, Jan. 1989.

[21] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, pages 323–351, 2005.

[22] D. M. Nicol. The cost of conservative synchronization in parallel discrete event simulations. *J. ACM*, 40(2):304–333, Apr. 1993.

[23] D. Rao and A. Chernyakhovsky. Parallel simulation of the global epidemiology of avian influenza. In *Simulation Conference, 2008. WSC 2008. Winter*, pages 1583 –1591, dec. 2008.

[24] G. F. Riley, M. H. Ammar, R. M. Fujimoto, A. Park, K. Perumalla, and D. Xu. A federated approach to distributed network simulation. *ACM Trans. Model. Comput. Simul.*, 14(2):116–148, Apr. 2004.

[25] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524, 2003.

[26] W. Small and J. D. Singer. Resort to arms: International and civil wars, 1816-1980. *Notices of the American Mathematical Society*, 1982.

[27] K. Soramaki, M. L. Bech, J. Arnold, R. J. Glass, and W. E. Beyeler. The topology of interbank payment flows. *Physica A: Statistical Mechanics and its Applications*, 379(1):317 – 333, 2007.

[28] W. Su and C. L. Seitz. Variants of the chandy-misra-bryant distributed discrete-event simulation algorithm. Technical report, Pasadena, CA, USA, 1988.

[29] S. Thulasidasan and S. Eidenbenz. Accelerating traffic microsimulations: A parallel discrete-event queue-based approach for speed and scale. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 2457 –2466, dec. 2009.

[30] X. F. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6 – 20, 2003.

[31] F. Wieland. Practical parallel simulation applied to aviation modeling. In *Proceedings of the fifteenth workshop on Parallel and distributed simulation*, PADS '01, pages 109–116, Washington, DC, USA, 2001. IEEE Computer Society.

[32] W. Willinger, D. Alderson, and J. C. Doyle. Mathematics and the internet: A source of enormous confusion and great potential, 2009.

[33] L. Zhang, X. Deng, J. Yu, and X. Wu. The degree and connectivity of internet's scale-free topology. *CoRR*, abs/1101.4285, 2011.