# A Performance Evaluation of WS-MDS in the Globus Toolkit

Ioan Raicu[*] Catalin Dumitrescu[*] Ian Foster[+*]

[*]*Computer Science Department*
*The University of Chicago*
*{iraicu,cldumitr}@cs.uchicago.edu*

[+]*Mathematics and Computer Science Division*
*Argonne National Laboratory*
*foster@cs.uchicago.edu*

## Abstract

*The Globus Toolkit® (GT) is the "de facto standard" for grid computing. Measuring the performance of various components of the GT in a wide area network (WAN) as well as a local area network (LAN) is essential in understanding the performance that is to be expected from the GT in a realistic deployment in a distributed and heterogeneous environment where there might be complex interactions between network connectivity and service performance. The focus of this paper is the performance of Monitoring and Discovery System (MDS), an essential GT component that is very likely to be used in a mixed; LAN and WAN environment. We specifically tested the scalability, performance, and fairness of the WS-MDS Index bundled with GT 3.9.5. To drive our empirical evaluation of WS-MDS, we used DiPerF, a DIstributed PERformance testing Framework, whose design was aimed at simplifying and automating service performance evaluation.*

## 1 Introduction

The Globus Toolkit [1, 2] is the "de facto standard" for grid computing as it has been called by numerous agencies and world recognized news sources such as the New York Times. Measuring the performance of various components of the Globus Toolkit is important to ensure that the Grid will continue to grow and scale as the user base and infrastructure expands. Furthermore, the testing of the toolkit and grid services is difficult due to the distributed and heterogeneous environment Grids are usually found in. Performing distributed measurements is not a trivial task, due to difficulties 1) *accuracy* – synchronizing the time across an entire system that might have large communication latencies, 2) *flexibility* – in heterogeneity normally found in WAN environments and the need to access large number of resources, 3) *scalability* – the coordination of large amounts of resources, and 4) *performance* – the need to process a large number of transactions per second.

In attempting to address these four issues, in previous work [3, 4] we developed DiPerF, a DIstributed PERformance testing Framework, aimed at simplifying and automating service performance evaluation. DiPerF coordinates a distributed pool of machines that run clients of a target service, collects and aggregates performance metrics, and generates performance statistics. DiPerF can be used to test the scalability and performance limits of a service: that is, find the maximum offered load supported by the service while still serving requests with an acceptable quality of service. Actual service performance experienced by heterogeneous and geographically distributed clients with different levels of connectivity cannot be easily gauged based on controlled LAN-based tests. Therefore significant effort is sometimes required in deploying the testing platform itself. With a wide-area, heterogeneous deployment provided by the PlanetLab [5, 6] and Grid3 [7, 8] testbed, DiPerF can provide accurate estimation of the service performance as experienced by such clients.

The focus of this paper is the performance of Monitoring and Discovery System (MDS), an essential Globus Toolkit component that is very likely to be used in a mixed, LAN and WAN environment. We specifically tested the scalability, performance, and fairness of the WS-MDS Index bundled with GT 3.9.5.

The contributions of this paper is a detailed empirical performance analysis of the WS-MDS component in the Globus Toolkit 3.9.5. Through our tests performed on WS-MDS, we have been able to quantify the performance gain or loss between various different implementations, and have found the upper limit on both scalability and performance on these services. We have also been able to show the performance of these components in a WAN, a task that would have been very tedious and time consuming without a tool such as DiPerF. By pushing the Globus Toolkit to the limit in both performance and scalability, we was able to give the users a rough overview of the performance they are to expect so they can do better resource planning. The developers also gained feedback on the behavior of the WS-MDS under heavy stress and allowed them to concentrate on improving the parts that needed the most improvements.

## 2 Related Work

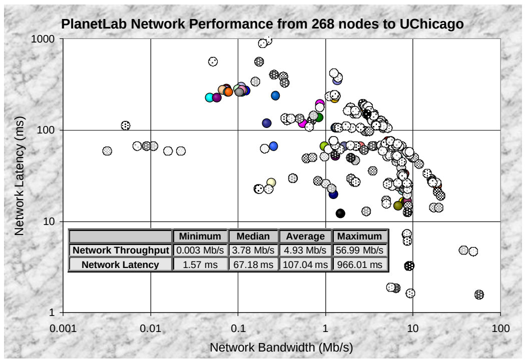
### 2.1 *MDS Performance Studies Related Work*

Zhang et al. [9] compared the performance of three resource selection and monitoring services: the Globus Monitoring and Discovery Service (MDS), the European Data Grid Relational Grid Monitoring Architecture (R-GMA) and Hawkeye. Their experiment uses two sets of machines (one running the service itself and one running clients) in a LAN environment. The setup is manual and each client node simulates 10 users accessing the service. This is exactly the scenario where DiPerF would have proved its usefulness: it would have freed the authors from deploying clients, coordinating them, and collecting performance results, and would allow focusing on optimally configuring and deploying the services to test, and on interpreting performance results. In a later study, Zhang et al. [10] investigated the performance of MDS using NetLogger and improved the testbed by having a cluster of 7 machines to handle the server side services and a cluster of 20 machines to handle the client side testbed; unfortunately, their study only addressed the performance of MDS in a LAN environment. On the other hand, our study of MDS was on a larger scale, utilizing over 100 machines in a WAN environment. We also tested the latest release of MDS (WS-MDS), based on the Globus Toolkit 3.9.5, which is the latest MDS implementation based on web services (WS). Aloisio et al. [11] studied the capabilities and limitations of the Globus Toolkit's Monitoring and Discovery Service; however, their experiments were limited to simple tests on a Grid Index Information Service (GIIS) only. Other Globus Toolkit component (i.e. GRAM, GridFTP [12], etc) performance evaluations have been done by the Globus Alliance as well. [13]

### 2.2 *Background Information*

#### 2.2.1 Testbeds

**Planetlab:** PlanetLab [14] is a geographically distributed platform for deploying, evaluating, and accessing planetary-scale network services. PlanetLab is a shared community effort by a large international group of researchers, each of whom gets access to one or more isolated "slices" of PlanetLab's global resources via a concept called distributed virtualization. PlanetLab's deployment is now at over 500 nodes (Linux-based PCs or servers connected to the PlanetLab overlay network) distributed around the world. Almost all nodes in PlanetLab are connected via 10 Mb/s network links (with 100Mb/s on several nodes), have processors speeds exceeding 1.0 GHz IA32 PIII class processor, and at least 512 MB RAM. Due to the large geographic distribution (the entire world) among PlanetLab nodes, network latencies and achieved bandwidth varies greatly from node to node. In order to capture this variation in network performance, Figure 1 displays the network performance between 268 nodes to 1 node at UChicago. It is very interesting to note the heavy dependency between high bandwidth / low latencies and low bandwidth / high latencies.



| | Minimum | Median | Average | Maximum |
|---|---|---|---|---|
| Network Throughput | 0.003 Mb/s | 3.78 Mb/s | 4.93 Mb/s | 56.99 Mb/s |
| Network Latency | 1.57 ms | 67.18 ms | 107.04 ms | 966.01 ms |

Figure 1: PlanetLab Network Performance from 268 nodes to a node at UChicago as measured by IPERF on April 13th, 2005; each circle denotes a node with the corresponding x-axis and y-axis values as its network characteristics, namely network latency and bandwidth, in log scale.

**UChicago CS Cluster:** The University of Chicago CS cluster contains over 100 machines that are remotely accessible. The majority of these machines are running Debian Linux 3.0, have AMD Athlon XP Processors at 2.1GHz, have 512 MB of RAM, and are connected via a 100 Mb/s Fast Ethernet switched network. The communication latency between any pair of machines in the CS cluster is on average less than 1 ms, with a few having latencies as high as several ms. Furthermore, all machines share a common file system via NFS (Network File System).

**DiPerF Cluster:** Some tests were performed on a smaller scale LAN that had better network connectivity, specifically 1Gb/s connections via a switch. The network latencies incurred were generally less than 0.1 ms. This cluster did not run NFS as was the case in the UChicago CS cluster. The connectivity of the DiPerF cluster to the outside world is 100Mb/s. The processor speeds ranged from 1.46GHz to 2.16GHz with 1GB of RAM on each node. One machine named "m5" had dual AMD Opteron 1.6GHz processors and was used to run the WS-MDS Index.

#### 2.2.2 WS-MDS

Nine GT4 services implement Web services (WS) interfaces: 1) job management (GRAM) [15]; 2) reliable file transfer (RFT); 3) delegation; 4-6) Monitoring and Discovery System (MDS) [16]; 7) community authorization (CAS); 8) OGSA-DAI data access and integration; 9) GTCP Grid TeleControl Protocol for online control of instrumentation.

The Monitoring and Discovery System (MDS), the focus of this paper, is composed of three components: MDS-Index, MDS-Trigger, and the MDS Aggregator. [16]

The Index Service collects monitoring and discovery information from Grid resources, and publishes it in a single location; generally, it is expected that a virtual organization will deploy one or more index services which will collect data on all of the Grid resources available within that virtual organization.

The Trigger Service collects data from resources on the grid and, if administrator defined rules match, can perform various actions. An example use is to send email when queue length on a compute resource goes over a threshold value.

The WS MDS Aggregator is the software framework on which WS MDS services (currently, the WS MDS Index and WS MDS Trigger services) are built. The aggregator framework collects data from an aggregator source and sends that data to an aggregator sink for processing. Aggregator sources distributed with the Globus Toolkit include modules that query service data, acquire data through subscription/notification, and execute programs to generate data. Aggregator sinks include modules that implement the WS MDS Index service interface and the WS MDS Trigger service interface.

### 2.2.3 DiPerF Framework

DiPerF [3, 4] is a DIstributed PERformance testing Framework aimed at simplifying and automating service performance evaluation. DiPerF coordinates a pool of machines that test a single or distributed target service, collects and aggregates performance metrics from the client point of view, and generates performance statistics. The aggregate data collected provides information on service throughput, service response time, on service 'fairness' when serving multiple clients concurrently, and on the impact of network latency on service performance. All steps involved in this process are automated, including dynamic deployment of a service and its clients, testing, data collection, and data analysis.

DiPerF consists of two major components: the *controller* and the *testers*. A user of the framework provides to the controller the address or addresses of the target service to be evaluated and the client code for the service. The controller starts and coordinates a performance evaluation experiment: it receives the client code, distributes it to testers, coordinates their activity, collects and finally aggregates their performance measurements. Each tester runs the client code on its machine, and gets periodic performance metrics from the client code made against the target service through a predefined interface. Finally, the controller collects all the measurements from the testers and performs additional operations (e.g., reconciling time stamps from various testers) to compute aggregated performance views.

The framework is supplied with a set of candidate nodes, and selects those available as testers. In future work, we will extend the framework to select a subset of available tester nodes to satisfy specific requirements in terms of link bandwidth, latency, compute power, available memory, and/or processor load. In its current version, DiPerF assumes that the target service is already deployed and running.

Some metrics are collected directly by the testers (e.g., response time), while others are computed at the controller (e.g., throughput and service fairness). Additional metrics (e.g. network related metrics, time to complete a subtask, etc), measured by clients can be reported, through an additional interface, to the testers and eventually back to controller for statistical analysis.

Communication among DiPerF components has been implemented with several flavors: ssh based, TCP, and UDP. When a client fails, we rely on the underlying protocols (i.e. whatever the client uses such as TCP, UDP, HTTP, pre-WS GRAM, etc) to signal an error which is captured by the tester which is in turn sent to the controller to delete the client from the list of the performance metric reporters. A client could fail because of various reasons: 1) predefined timeout which the tester enforces, 2) client fails to start (i.e. out of memory - OS client machine related), 3) and service denied or service not found (service machine related). Once the tester is disconnected from the controller, it stops the testing process to avoid loading the target service with requests which will not be aggregated in the final results.

## 3 Empirical Performance Results

### *3.1 WS-MDS Index Performance*

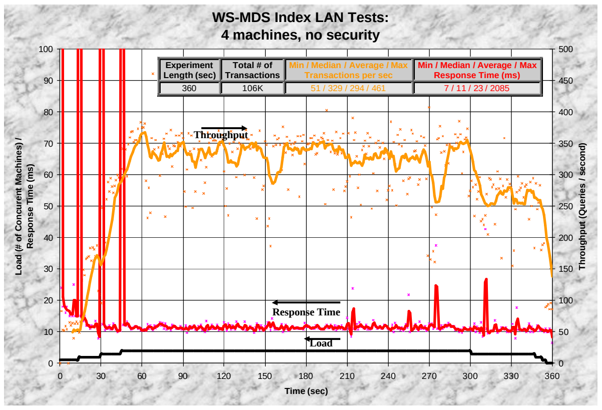
We evaluated the WS-MDS index bundled with the Globus Toolkit 3.9.5 on a machine at UChicago. The metrics collected (client view) by DiPerF are:

- **Service response time** or time to serve a query request, that is, the time from when a client issues a request to when the request is completed
- **Service throughput**: aggregate number of queries per second from the client view
- **load**: number of concurrent service requests

We ran our experiments on four different configurations: 1) LAN tests with 4 machines connected via 1 Gb/s from the DiPerF cluster; 2) WAN tests with 128 machines from PlanetLab connected via 10 Mb/s; 3) WAN tests with 288 machines from PlanetLab connected via 10 Mb/s; and 4) LAN+WAN tests with 3 machines in a LAN and 97 machines in a WAN. We ran the WS-MDS index at UChicago on a dual AMD Opteron 1.6GHz with 1GB RAM, 1Gb/s network connection locally, and 100 Mb/s network connection externally; the DiPerF framework ran on a similarly configured machine, except that it had a single AMD K7 processor running at 2.16GHz. For each set of tests, the caption below the figure will address the particular configuration of the DiPerF controller and the testbed configuration which yielded the respective results.

In the figures to follow (Figure 2 – Figure 9), each series of points representing a particular metric is also approximated using a moving average over a 60 point interval; since most raw samples were taken once a second, the moving average represents the average over 1 minute intervals. Each figure also has a summary of the results in a table in the upper right hand corner of each figure; the table contains the experiment length in time, number of queries processed by the WS-MDS Index, and the minimum/median/average/maximum of both collected metrics, namely throughput (queries per second), and response time (ms).

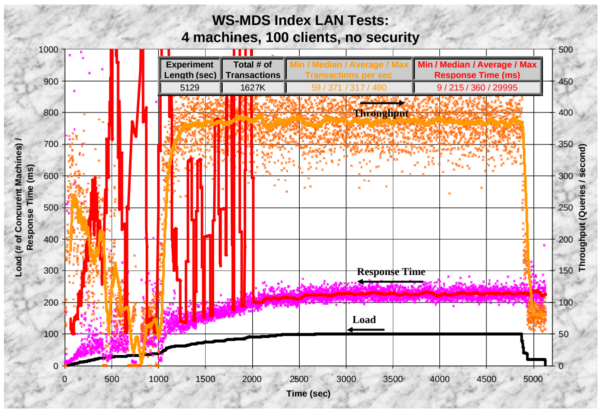
Figure 2 represents the performance of the WS-MDS index with no security when 4 clients in a LAN environment targeted the WS-MDS index. Note the large spikes in response times at the beginning of the experiment; each spike corresponds to a new client starting, and they generally occur only for the first query; the summary table gives us a rough idea of how long these peaks were, with the highest being over 2 seconds. The throughput in terms of transactions per second reached as high as 461 queries per second, although for the majority of the experiment it was mostly between 300 and 350 queries per second. Each of the four clients could generate between 80 to 130 queries per second, so the aggregate performance along with the fact that the processors on the WS-MDS Index were only utilized about 60%, it leads us to believe that the WS-MDS index could achieve even higher throughput. The network throughput we observed seemed to be between 8Mb/s and 12Mb/s at the WS-MDS Index, so we do not believe that the network capacity of 1Gb/s was an issue. The same exact experiment produced very similar results even when the LAN network connectivity was downgraded to 100Mb/s.



| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 360 | 106K | 51 / 329 / 294 / 461 | 7 / 11 / 23 / 2085 |

Figure 2: WS-MDS Index LAN Tests with no security including 4 clients running on 4 physical nodes at UChicago in a LAN connected via 1 Gb/s links; tunable parameters: utilized 4 concurrent clients, with each client starting every 15 seconds; left axis – load, response time; right axis – throughput

Figure 3 represents a very similar test to Figure 2, except that we ran 100 clients over the same 4 machines in a LAN. Note the same behavior in response times at the beginning of the experiment, in which there are very large spikes in response times as new clients join.

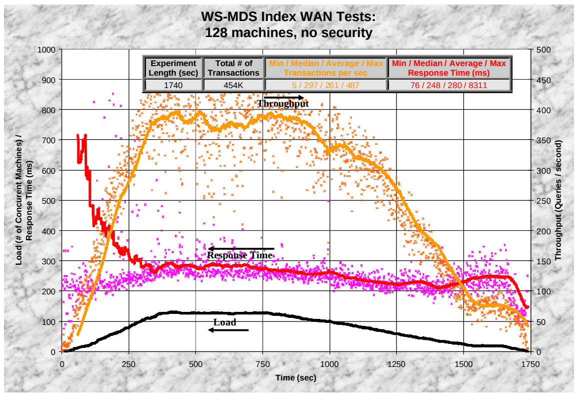


| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 5129 | 1627K | 59 / 371 / 317 / 490 | 9 / 215 / 360 / 29995 |

Figure 3: WS-MDS Index LAN Tests with no security including 100 clients running on 4 physical nodes at UChicago in a LAN connected via 1 Gb/s links; tunable parameters: utilized 100 concurrent clients, with each client starting every 15 seconds

The new clients joining (at a rate of 3 per second) seem to significantly affect the ability of the WS-MDS Index to process queries; the throughput starts out strong, but it starts to drop almost reaching 0 in the beginning period in which many clients are joining in the experiment. During the period of low throughput due to new clients joining, the CPU utilization was very low, with instances when it would reach an idle state for entire seconds at a time. Otherwise, the only other difference between this experiment and the one in Figure 2 is that the throughput reached almost 500 queries per second with an average of just below 400 transactions per second. The CPU utilization was closer to saturation with 80~90% utilization. The network usage was around 12Mb/s and 15Mb/s.

Figure 4 represents a larger experiment including 128 nodes from PlanetLab in a WAN environment that has network bandwidth connectivity of 10Mb/s and network latencies of 60 ms on average, and as high as 200 ms on some nodes.

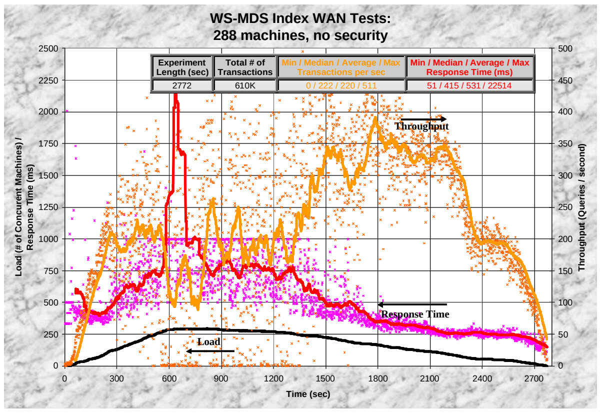


| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 1740 | 454K | 5 / 297 / 261 / 487 | 76 / 248 / 280 / 8311 |

Figure 4: WS-MDS Index WAN Tests with no security including 128 clients running on 128 physical nodes in PlanetLab in a WAN connected via 10 Mb/s links; tunable parameters: utilized 128 concurrent clients, with each client starting every 3 seconds

The maximum throughput achieved was just under 500 queries per second, with close to 400 queries per second on average during the peak portion of the experiment when all 128 clients were accessing the WS-MDS Index simultaneously. It is interesting to note that the response time only increased from the low 200 ms to the high 200 ms. The response time during the peak portion of the experiment when there were 128 machines actively querying the WS-MDS Index seems very similar (248 ms vs. 215 ms) to the performance of the same experiment performed in a LAN as presented in Figure 3; the 15% longer response times in the WAN tests could be attributed simply to the fact that there were 30% more clients actively participating. We observed that the CPU utilization and network bandwidth usage were similar to the LAN tests from Figure 3.

Figure 5 represents another experiment utilizing PlanetLab, but this time we used 288 machines all over the world instead of the 128 machines from the USA.



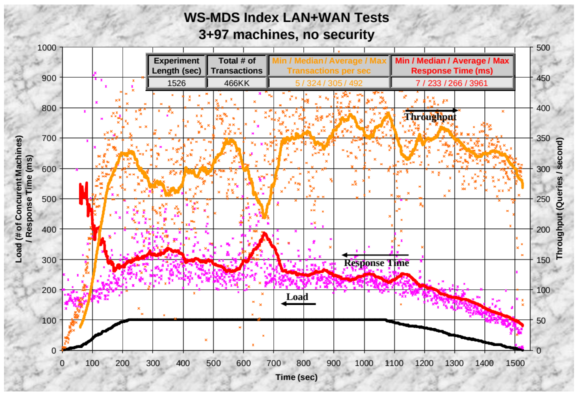
| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 2772 | 610K | 0 / 222 / 220 / 511 | 51 / 415 / 531 / 22514 |

Figure 5: WS-MDS Index WAN Tests with no security including 288 clients running on 288 physical nodes in PlanetLab in a WAN connected via 10 Mb/s links; tunable parameters: utilized 288 concurrent clients, with each client starting every 2 seconds

This test was very interesting due to the fact that the throughput achieved while all 288 machines were concurrently accessing the index was around 200 queries per second on average (when compared to almost 400 queries per second that we achieved from only 128 machines). As the number of machines started dropping, the throughput started to increase, and by about 200 clients, the throughput reached a level more similar to what we had seen in previous tests. Although the WS-MDS Index managed to service all the 288 clients concurrently, its efficiency in terms of sustaining a high throughput clearly dropped.

Figure 6 and Figure 7 represent the same experiment in which we used 97 machines from PlanetLab in a WAN setting, and 3 machines at UChicago in a LAN setting. Figure 2 and Figure 4 both showed that each LAN and WAN individually could achieve 300 to 400 queries per second independently. Figure 5 did not show any improvement in the achieved throughput (it actually decreased), from which we concluded that the index could efficiently handle so many clients in a WAN environment. Figure 6 and Figure 7 tries to capture the peak performance of the WS-MDS Index with a testbed that we know could generate more queries per second than we actually observed in this experiment. Ultimately, we obtained performance similar to that of Figure 3 and Figure 4 in which we had 100 clients in a LAN and 128 clients in a WAN respectively. One of our conclusions after these series of tests is that the

WS-MDS Index peak throughput on the specific hardware and OS we ran it on is 500 queries per second. Our second conclusion is that a WAN environment can achieve comparable numbers to that of LAN environments given a large enough pool of machines. On the other hand, at least in a WAN environment, it seems that the efficiency of the index decreases after a critical mass of clients is reached; this critical number of clients seems to be in the area of 100 to 200 clients for our particular hardware that we ran the WS-MDS index on and the specific characteristics of the testbed utilized.



| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 1526 | 466KK | 5 / 324 / 305 / 492 | 7 / 233 / 266 / 3961 |

Figure 6: WS-MDS Index LAN+WAN Tests with no security including 100 clients running on 3 physical nodes at UChicago (LAN) and 97 physical nodes in PlanetLab (WAN); tunable parameters: utilized 100 concurrent clients, with each client starting every 2 seconds

The next interesting thing we wanted to extract from this experiment was how much did the WAN clients contribute towards the overall achieved throughput and how much we could attribute to the LAN clients. Just as a reminder, the 97 PlanetLab WAN clients were connected via 10Mb/s links with network latencies between 20 ms and 200 ms and an average of 60 ms. On the other hand, the 3 LAN clients were connected via 1Gb/s links and 0.1 ms network latencies. Figure 7 represents our findings about how much the LAN clients and the WAN clients each contributed towards the overall throughput.

Figure 7 shows that for the peak portion of the experiment when all 100 clients were running in parallel, the LAN clients (3% of all clients) generated 5% of the queries. During this period, the 3 LAN clients generated on average 18 queries per second, while in Figure 2 we observed that 4 LAN clients could achieve a throughput well over 350 queries per second. Note how the LAN throughput increases as the WAN clients stop participating, eventually reaching close to 500 queries per second.

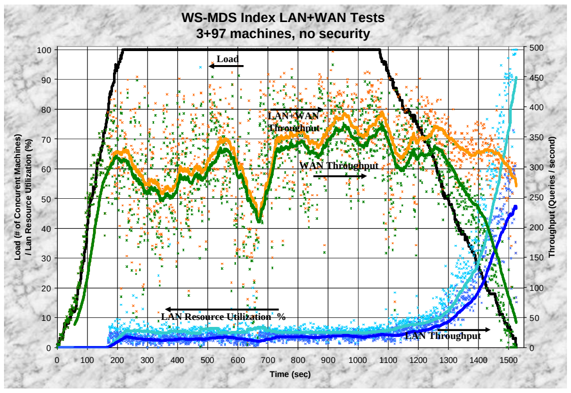
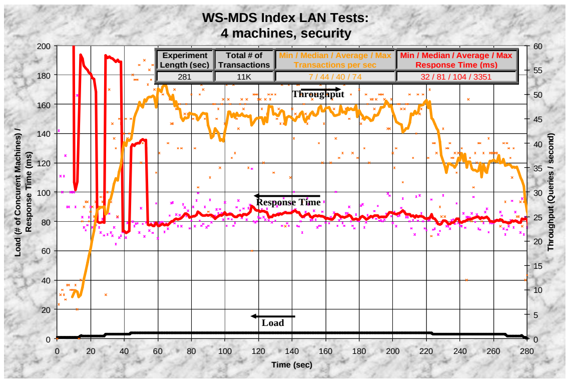


Figure 7: WS-MDS Index LAN+WAN Tests with no security including 100 clients running on 3 physical nodes at UChicago (LAN) and 97 physical nodes in PlanetLab (WAN); tunable parameters: utilized 100 concurrent clients, with each client starting every 2 seconds

This behavior is very peculiar, and it shows the effects of a particular design choice of the developers. Based on the results of these experiments, we concluded that under heavy load, the WS-MDS Index acts as round robin queue. This caused all clients (irrespective of the fact that some were very well connected while others were not) to get equal share of resources. We also observed that the achievable throughput is lower when there are new clients joining. The initial connection is expensive on the service side that it affects the index's capacity to process queries; there seem to be some concurrency issues especially since the processors do not seem heavily loaded in the beginning part of the experiments when many clients were joining the experiment, although the achieved throughput was relatively low.

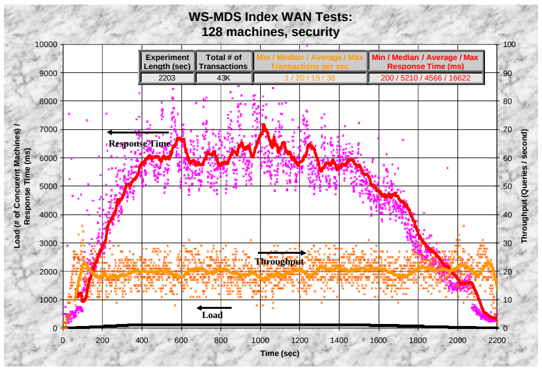
Both Figure 8 and Figure 9 cover experiments against the WS-MDS Index with security enabled.



| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 281 | 11K | 7 / 44 / 40 / 74 | 32 / 81 / 104 / 3351 |

Figure 8: WS-MDS Index LAN Tests with security including 4 clients running on 4 physical nodes at UChicago in a LAN connected via 1 Gb/s links; tunable parameters: utilized 4 concurrent clients, with each client starting every 15 seconds

Figure 8 was a test performed in a LAN environment which achieved a throughput of 45 queries per second generated by 4 clients at UChicago. Note the response times of about 80 ms per query. Unlike the results from the WAN vs. LAN tests without security in which we obtained similar throughput capacity of the index, the WAN tests with 128 clients proved to have significantly lower performance. The WAN test resulted in a throughput of about 20 queries per second, and with response times on the order of 5 to 7 seconds. Overall, adding security to the WS-MDS Index queries incurs a significant performance penalty.



| Experiment Length (sec) | Total # of Transactions | Min / Median / Average / Max Transactions per sec | Min / Median / Average / Max Response Time (ms) |
|---|---|---|---|
| 2203 | 43K | 1 / 20 / 19 / 36 | 200 / 5210 / 4566 / 16622 |

Figure 9: WS-MDS Index WAN Tests with security including 128 clients running on 128 physical nodes in PlanetLab in a WAN connected via 10 Mb/s links; tunable parameters: utilized 128 concurrent clients, with each client starting every 3 seconds

### 3.2 *WS-MDS Index "Fairness"*

In order to visualize the "fairness" of the resource sharing among the various MDS clients, we took a subset (only the peak in which all 288 clients were concurrently generating queries) of the experiment depicted in Figure 5 and plotted (Figure 10) the number of successful queries (the size of the bubble) against the nodes network connectivity.

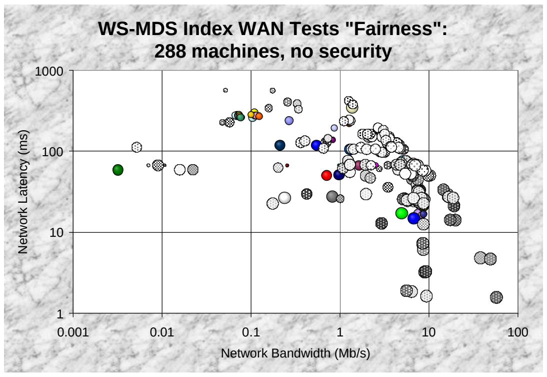


Figure 10: "Fairness" results for the WS-MDS Index WAN Tests with no security including 288 clients running on 288 physical nodes in PlanetLab

We see how the majority of the nodes seem to be relatively the same size, especially for the well connected ones. As the latency increases, we start noticing an increasing number of smaller bubbles, which shows that poorly connected nodes were getting a smaller share of resources. Overall, despite the large variance in network connectivity, the results from Figure 10 show a relatively fair distribution of resources among the 288 clients distributed throughout the world.

## 4 Conclusion

In this paper, we have used DiPerF, a DIstributed PERformance testing Framework, to analyze the performance scalability of the WS-MDS Index bundled with the Globus Toolkit 3.9.5. We measured the performance in a wide area network (WAN) as well as a local area network (LAN) with the goal of understanding the performance that is to be expected from the Globus Toolkit in a realistic deployment in a distributed and heterogeneous environment.

Through our tests performed on the WS-MDS, we have been able to quantify the performance gain or loss between various different versions or implementations, and have normally found the upper limit on both scalability and performance on these services. We have also been able to show the performance of these components in a WAN, a task that would have been very tedious and time consuming without a tool such as DiPerF. By pushing the Globus Toolkit to the limit in both performance and scalability, we were able to give the users a rough overview of the performance they are to expect so they can do better resource planning. The developers also gained feedback on the behavior of the various components under heavy stress and allowed them to concentrate on improving the parts that needed the most improvements.

### 4.1 *WS-MDS Results Summary*

Table 1 shows the summary of the main results from the WS-MDS experiments performed in LAN and WAN with both security enabled and disabled.

With no security, WAN throughput performance was similar to that of LAN throughput performance given a large enough set of clients. On the other hand, with security enabled, the WAN tests showed less than half the throughput when compared to a similar test from a LAN. The initial query for a connection could take a very long time (we observed times as high as 30 seconds, and was almost always greater than a second); furthermore, many new connections adversely affects the efficiency of the index and its ability to process as many queries as it could have if it weren't for the new connections. During the start-up phase of the experiments when many clients would be making new connections, the throughput would be relatively poor, and the CPU utilization would be low as well. Another observation is that under heavy load, the WS-MDS index acts as a round robin queue which effectively distributes the share of resources evenly across all clients irrespective of the connectivity of the client.

Table 1: WS-MDS summary of experiments in both LAN and WAN with both security enabled and disabled; for the "load at service saturation, the * indicates that the service was not saturated with the number of concurrent clients that were used

| Experiment Description | Throughput (trans/sec) | | | | Load at Service Saturation | Response Time (ms) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Med | Aver | Max | | Min | Med | Aver | Max |
| Figure 50: LAN 4 | 51 | 329 | 294 | 461 | 4* | 7 | 11 | 23 | 2085 |
| Figure 51: LAN 4-100 | 59 | 371 | 317 | 490 | 60 | 9 | 215 | 360 | 29995 |
| Figure 52: WAN 128 | 5 | 297 | 261 | 487 | 128* | 76 | 248 | 280 | 8311 |
| Figure 53: WAN 288 | 0 | 222 | 220 | 511 | 225 | 51 | 415 | 531 | 22514 |
| Figure 54: WAN 97 + LAN 3 | 5 | 324 | 305 | 492 | 100* | 7 | 233 | 266 | 3961 |
| Figure 56: LAN 4 + security | 7 | 44 | 40 | 74 | 4* | 32 | 81 | 104 | 3351 |
| Figure 57: WAN 128 + security | 1 | 20 | 19 | 36 | 22 | 200 | 5210 | 4566 | 16622 |

### *4.2 Lessons Learned*

Some lessons we learned through the work presented in this thesis are:

- building scalable software is not a trivial task
- C-based components of the Globus Toolkit normally perform better than their Java counterparts
- depending on the particular service tested, WAN performance is not always comparable to that found in a LAN; for example, WS-MDS with no security performed comparable between LAN and WAN tests, but WS-MDS with security enabled achieved less than half the throughput in a WAN when compared to the same test in a LAN
- the testbed performance (in our case it was mostly PlanetLab) can influence the performance results, and hence careful care must be taken in comparing experiments done at different times when the state of PlanetLab could have significantly changed

We conclude with the thought that we succefully tested the WS-MDS Index performance in a variety of configurations from a LAN to a WAN environment to having security enabled and disabled. We have shown in previous work [4] DiPerF's accuracy as being very good with only a few percent of performance deviation between the aggregate client view and the centralized service view. Essentially, we have contributed towards a better understanding of a very important and vital Globus Toolkit component, namely WS-MDS.

## 5 Bibliography

[1] The Globus Alliance, www.globus.org.
[2] "GT4 Release Contents". http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/GT4Facts/index.html#Contents
[3] C. Dumitrescu, I. Raicu, M. Ripeanu, I. Foster. "DiPerF: an automated DIstributed PERformance testing Framework." 5th International IEEE/ACM Workshop in Grid Computing, 2004, Pittsburg, PA.
[4] I. Raicu. "A Performance Study of the Globus Toolkit® and Grid Services via DiPerF, an automated DIstributed PERformance testing Framework," MS Thesis, Department of Computer Science, University of Chicago, May 2005.
[5] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet", Proceedings of the First ACM Workshop on Hot Topics in Networking (HotNets), October 2002.
[6] A. Bavier et al., "Operating System Support for Planetary-Scale Services", Proceedings of the First Symposium on Network Systems Design and Implementation (NSDI), March 2004.
[7] I. Foster, et al., "The Grid2003 Production Grid: Principles and Practice", 13th IEEE Intl. Symposium on High Performance Distributed Computing, 2004.
[8] Grid3. http://www.ivdgl.org/grid3/
[9] X. Zhang, J. Freschl, and J. Schopf. "A Performance Study of Monitoring and Information Services for Distributed Systems." Proceedings of HPDC, August 2003.
[10] X. Zhang and J. Schopf. "Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2." Proceedings of the International Workshop on Middleware Performance (MP 2004), April 2004.
[11] G. Aloisio, M. Cafaro, I. Epicoco, and S. Fiore, "Analysis of the Globus Toolkit Grid Information Service". Technical report GridLab-10-D.1-0001-GIS_Analysis, GridLab project.
[12] GridFTP: Universal Data Transfer for the Grid. Globus Project, White Paper.
[13] The Globus Alliance, "Overview and Status of Current GT Performance Studies", http://www-unix.globus.org/toolkit/docs/development/3.9.5/perf_overview.html
[14] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," ACM Computer Communications Review, July 2003.
[15] "GT 4.0 WS GRAM Approach". http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/execution/key/WS_GRAM_Approach.html
[16] "GT 4.0 Component Fact Sheet: WS MDS". http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/info/WSMDSFacts.html