AI based Performance Benchmarking & Analysis of Big Data and Cloud Powered Applications

An in Depth View

Jayanti Vemulapati†
Accenture Digital
Accenture Solutions Pvt Ltd
Hyderabad, Telangana, India
jayanti.vemulapati@accenture.com

Anuruddha S. Khastgir
Accenture Digital
Accenture Solutions Pvt Ltd
Mumbai, Maharashtra, India
anuruddha.s.khastgir@accenture.com

Chethana Savalgi
Accenture Digital
Accenture Solutions Pvt Ltd
Bengaluru, Karnataka, India
chethana.savalgi@accenture.com

ABSTRACT

Big data analytics platforms on cloud are becoming mainstream technology enabling cost-effective rapid deployment of customer's Big Data applications delivering quicker insights from their data. It is, therefore, even more imperative that we have high performant platform infrastructure and application at a reasonable cost. This is only possible if we make a transition from traditional approach to execute and measure performance by adopting new AI techniques such as Machine Learning (ML) & predictive approach to performance benchmarking for every application domain.

This paper proposes a high-level conceptual model for automated performance benchmarking which includes execution engine that has been designed to support a self-service model covering automated benchmarking in every application domain. The automated engine is supported by performance scaling recommendations via prescriptive analytics from real performance data set.

We furthermore extended the recommendation capabilities of our self-service automated engine by introducing predictive analytics for making it more flexible in addressing 'what-if' scenarios to predict 'Right Scale' with measurement of "Performance Cost Ratio" (PCR). Finally, we also present some real-world industry examples which have seen the performance benefits in their applications with the recommendations given by our proposed model.

CCS CONCEPTS

• General and reference~Performance • Software and its engineering~Software performance • Software and its engineering~Cloud computing

†Corresponding Author

KEYWORDS

Auto Scale; Right Scale; AI; ML; Benchmarking; Automation; Performance Cost ratio; Predictive Analytics; Scale Factor; Performance Metrics; Big Data; Complex Deployments; Performance tuning

ACM Reference format:
Jayanti Vemulapati, Anuruddha Khastgir and Chethana Savalgi. 2019 AI based Performance Benchmarking & Analysis of Big Data and Cloud Powered Applications: An in Depth View. *In Proceedings of Tenth ACM/SPEC International Conference on Performance Engineering, (ICPE'19), April 7–11, 2019, Mumbai, India*, ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3297663.3309676

1 WHAT IS PERFORMANCE BENCHMARKING?

Performance Benchmarking can be defined as a process to measure or gauge performance of an application or a system in terms of processing time/response time and throughput vis-à-vis the default system which is not tuned or optimized. It also covers the re-measuring of the system/application performance when it has been tuned/optimized at the infrastructure and application service levels.

Performance benchmarking has at-least two folds benefits; foremost one helps in getting the best performance from existing resources and applications, and second one largely helps in defining the below aspects for applications or services:

- 1. Set performance metrics baselines for Service Level Agreement (SLA)
- 2. Compare your service against competitors
- 3. Offering at par with the industry standards and become a high performant service
- 4. Repository of reusable and highly performant PaaS use-cases and SaaS re-usable patterns
-
- 5. Help in high performant service enablement
- 6. Reduction in Mean Time between Failures (MTBF) and Mean Time to Repair (MTTR)

2 PERFORMANCE BENCHMARKING PARAMETERS AND METRICS

There are several parameters which drives the performance. The most critical one's are classified into the following categories:

- • @Infrastructure level: CPU processor type, Memory, disk type: Hard Disk Drive (HDD), or Solid-State Drive (SSD), network bandwidth, O/S, Virtualization type
- • @System resource allocation: % of the system resource (such as cores, memory allocation) allocated to the application to consume
- • @Application level: Data security like data encryption for application domain like Ecommerce etc. For Example, what is the performance over-head when data at rest and data in motion is encrypted

The main performance metrics to be measured are as follows:

- 1. CPU utilization
- 2. Memory Utilization
- 3. Disk Utilization
- 4. Network latency and throughput
- 5. Application metrics
  a. Processing time
  b. Throughput

3 PERFORMANCE BENCHMARKING MEASUREMENTS

The system resources metrics and application related metrics are measured and correlated to produce a value called Performance Cost Ratio (PCR). PCR is the basic unit of measurement for any performance benchmark activity.

Performance Cost Ratio (PCR)

PCR =PT/IC

PT: Total processing time in sec
IC: Infrastructure cost

We all know that infrastructure need of the customer applications changes as and when business grows, and this can be achieved with baselined PCR and the scale factor. More ever, this can be done seamlessly if we have auto scale feature and Right scale feature in place.

Auto scale: The ability to grow and shrink the infrastructure as per the application needs. This is supported by most of the cloud providers like AWS but not the Right Scale as explained below.

Right scale[1]: The ability to pick and choose the right node in terms of configuration (CPU cores and memory) and right number of nodes as per the application need and PCR (Performance Cost Ratio).

[1] This term is coined by the corresponding author of this paper.

We must constantly measure PCR per scenario as per the customer requirements. For e.g. some of the scenarios HDD storage can be recommended which is cheaper in cost and works well with the algorithms which do not require random seek of the disk. It also works well for in-memory processing where most of the computations are happening in memory.

4 PERFORMANCE BENCHMARKING METHODOLOGY

The methodology covers the complete benchmarking paradigm to produce the results as per industry standards. The step-wise execution methodology is as follows:

1. Understand business scenarios and classify them to right application domain
2. Data characterization by applying the Big Data 5V techniques (Volume, Variety, Velocity, Veracity, Value)
3. Analyze workloads and generate the data sets either real or synthetic datasets as per the industry standards like TPC [4]
4. Identify right set of tools to measure the performance benchmarking metrics
5. Setup the isolated environment and execute the runs (minimum 3 runs till the Relative Standard Deviation (RSD) is <= defined threshold)
6. Correlate the raw output from the tools with the system resource utilization and identify the hot spots [Ref. Table 3, Figure 3 & 4]
7. Tune and optimize the infrastructure and application services and re-run the execution
8. Final analysis and recommendations

5 PERFORMANCE BENCHMARKING TOOLS SELECTION

Tools are selected for each of the application domain on the basis of following features:

1. Support for different workload patterns
2. Ability to generate workload data sets
3. Maximum coverage of use-cases
4. Industry Standard

For example, for Big Data like Hadoop we use BigDataBench [2], for Cassandra we use YCSB [3] etc.

6 SELF-SERVICE PERFORMANCE BENCHMARKING WITH MACHINE LEARNING (ML) AND AUTOMATION

Performance Benchmarking of large-scale complex deployments like Big Data applications on cloud are limited by the traditional approach and methods of Benchmarking. It is a time-consuming process ranging from setting up of the environment with required tools, applying the right tuned configuration, execution of benchmarks and finally correlating the raw outputs to produce a meaningful insight. Hence ML based automation becomes necessity to cut down over all elapsed time and produce recommendations based on prescriptive and predictive analysis.

Moreover, vendor offered cloud based Big data analytics platforms are becoming mainstream technology enabling cost-effective rapid deployment of client's Big Data applications for delivering quicker insights from their data. In these environments, benchmarking and capacity planning are performed in close collaboration between vendors and customers. For customers wishing to benchmark their prospective Big Data analytics application, should be able to simply use pay-per-use performance benchmarking on-demand self-service models to deploy and benchmark their applications in the cloud. They may want to perform the following tasks:

- Browse (without any prior background information) on some of the recommendations on optimized set of nodes, concurrent users, Response/Processing time
- Execute performance benchmarks with scenario of their choice on the given platform
- Execute their applications in Auto Scale and Right Scale mode

The customers may want to do some or all the afore-mentioned self-service scenarios when migrating to the cloud or on a continuous basis to assess the performance of their applications in the cloud. One immediate benefit for customers would be to gain confidence in their applications running with best performance without requiring long setups, planning and costly operation. Also, this would enable the customers with making informed decisions to scale up/down with Auto Scale and Right Scale features.

We have designed and developed automated performance benchmarking execution engine with plugins for various tools so that the execution of the benchmarks on the system can run continuously without human interactions during isolated benchmarking hours such as off-peak hours identified for the production environments.

The plugin framework allows for easy harnessing of new test scenarios and workload patterns for automated execution and report generation. The scenario specific benchmarking tools are packaged into the execution engine in form of plugins. This makes the benchmarking execution engine adaptable to any specified Application Domain or workload. For instance, if Online Analytical Processing (OLAP) benchmarking is selected (or queried for), then the user has the option to choose from Cloudera Hadoop, Hortonworks Data Platform or Spark. The execution engine also helps us un-earth interesting patterns on response time variation on jobs executed on different period and this variation primarily can be attributed towards cloud virtualization.

Additionally, the self-service model monitors the system under benchmark in terms of performance metrics such as system processing time and throughput. The model applies Auto Scale/Right Scale features to maintain the performance metrics and also to meet the threshold performance levels as specified by the user.

Lastly in a self-service mode a Virtual (AI) (ref. Figure 1 below) assistant can be deployed to understand the end user requirements and guide them to execute their own bench marks for their environments.

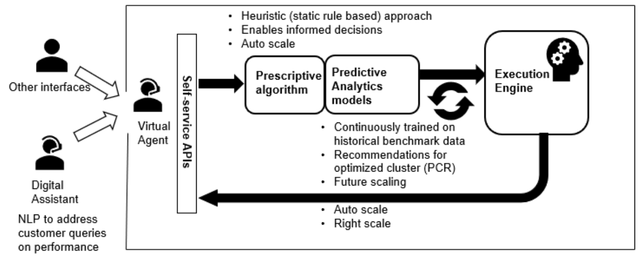
Heuristic (static rule based) approach
Enables informed decisions
Auto scale
Other interfaces
Virtual Agent
Self-service APIs
Prescriptive algorithm
Predictive Analytics models
Execution Engine
Digital Assistant
NLP to address customer queries on performance
Continuously trained on historical benchmark data
Recommendations for optimized cluster (PCR)
Future scaling
Auto scale
Right scale

Figure 1 Self-service model for performance benchmarking

7 PREDICTIVE ANALYTICS OVER PRESCRIPTIVE ANALYTICS

For the recommendations on optimized set of nodes, no. of concurrent users, response/processing time, we have taken predictive analytics approach (this initiative is currently work-in-progress) which would be a significant improvement over our already existing Prescriptive model (static rule based model), which is based on our extensive benchmarking experience on Big Data Analytics applications on cloud and rich performance data collected over the period. Since, we have a rich collection of performance data, it is only a logical transition to design and apply predictive analytics model on top of the collection of performance data set and make predictions on auto scale, right scale based on PCR to our clients. The prediction modelling approach is based on some of the following design principles:

- Performance (historical) data to build the model: Use variable data size: (500 GB, 1TB, 2TB, 5TB etc.). This is useful in providing correlation between effects of increasing data size on total processing time
- Consider complex workloads/queries (medium to complex mix, should be able to recommend ballpark range)

The model will calculate the recommended optimized set of nodes based on (predicted) no. of cores, RAM, total processing time and PCR.

As an example, the predictive model running in backend to our self-service interface should be able to answer the following question received from users: "If I need to process 1TB of data, with 10 concurrent users, running complex queries, response time <= 163 seconds, what should be my cluster size?"

The model be able to recommend optimized number of concurrent users supported ~ to be calculated based on heuristic (static) rules and also Scaling – Mostly horizontal (easy to scale up/down); In few cases – vertical scaling may be recommended when more relevant (depending on workload complexity, CPU, memory intensive workloads)

We take a simple and quick use case scenario (How many concurrent users at any point of time can my Hadoop cluster support?). To demonstrate how our automated performance benchmarking execution engine can help customers in determining the recommendation for optimized multi-users scaling and scaling threshold (maximum multi-users to run) on a given performance tuned Hadoop cluster.

We performed a simple Linear Regression analysis to find the correlation between multi user scenarios and processing time. The analysis was done on a given sample of performance data collected for OLAP complex queries on TPC-DS data [5] running on 7 data node tuned Hadoop cluster. The number of concurrent users run were 1, 2, 6, 8, 10 in the order of sequence.

The regression model thus derived out of the sample data was then used to do 'what-if' analysis for increased number of concurrent users, i.e. to see with what accuracy the model could predict total processing time for 12, 14 and 16 concurrent users. The predicted values were then compared vis-a-viz actual total processing time captured for 12, 14 and 16 concurrent users that were run on the same cluster. The analysis and results are provided in Figure 2 & Figure 3 below.

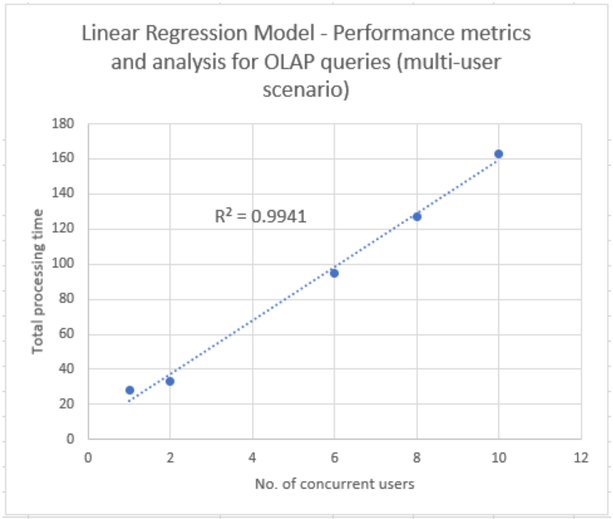

Figure 2 A Simple Linear Regression analysis of OLAP performance metrics against multi-user scenario

Table 1 Performance comparison – Actual versus Predicted

No. of concurrent users	Actual total processing time (seconds)	Predicted total Processing time (seconds)	Difference (Actual - Predicted) (seconds)
12	182.00	189.72	7.72
14	212.00	220.18	8.18
16	368.00	250.64	117.36

From the above Table 1, we observe that for 16 users, the total processing time is an outlier. After a deep analysis of the cluster tuning parameters, it was identified to be a queue wait period overhead that is getting added to the total processing time for queries when concurrency is increased beyond the given cluster capacity. Hence for illustration purpose, for our customer query discussed in the beginning of this section, our self-service automated performance benchmarking execution engine would return with a final recommendation (also taking into considerations other performance metrics such as %CPU and %RAM utilization) that 10 is the optimized number of concurrent users and 14 being the maximum concurrency the cluster size can support.

8 LEVERAGING MACHINE LEARNING TO ADDRESS COMPLEX USE CASES AND RECOMMENDATIONS

A slight variation to the use case dealt in the previous section would be if a customer is seeking recommendations on optimized multi-users scaling and scaling threshold (maximum multi-users to run) based on success rate of the workloads in a given performance tuned Hadoop cluster. For instance, a customer can ask, "what's the maximum number of concurrent users my Hadoop cluster can support with 100% success rate of the workloads?"

On the other hand, the customer could also ask for a recommended cluster size that would support maximum 30 concurrent users, running 60 OLAP queries of mixed complexities (medium to complex) with 100% success rate.

We performed an analysis and modeled the problem statement for a suitable ML algorithm to solve the problem. The analysis was done on a given sample of performance data collected for OLAP medium to complex queries running on 8 data-node Hadoop cluster that was tuned for performance. The concurrent users running the query were added in ramp-up mode during execution of the workloads. The maximum number of concurrent users run were 35. The workload designed was based on random load pattern of mixed queries - each user executed query which was selected randomly from an 8-query set. Once the query execution was completed for a user, the next query was selected randomly. Queries were executed on the

Hadoop cluster from the Gateway node. Queries were executed over connections established using JDBC driver and Beeline client.

Two different cluster configurations were used:

- Cluster Configuration 1: Total 976 GB memory and 128 CPU Cores comprising of 8 data nodes, data size of 500 GB with replication factor 3.
- Cluster Configuration 2: - Total 1952 GB memory and 256 CPU Cores comprising of 8 data nodes, data size of 500 GB with replication factor 3.

The participating variables used in the model (as outcome and predictor variables) were: Total processing time, number of concurrent users, total (cluster) CPU cores, Total (cluster) RAM size, service-based resource allocation (for example, yarn memory allocation for Spark), total queries run and success rate (total queries successfully run out of total queries run) for a certain number of concurrent users.

Support Vector Machine (SVM) classification model was trained with the (80%) of sample data (selected randomly) as supervised learning process. The SVM model thus derived out of the sample data was then run on remaining 20% of test data to see with what accuracy the model could predict success rates. Instead of classifying the outcome variable 'query success rate' between 100% success rate and less than 100% success rate, ranges of success rates were introduced (A, B, C, D and E) in the train dataset to keep the PCR more flexible.

For instance, a customer may allow for 90% of success rate of workloads to go for cheaper infrastructure cost option.

The sample dataset structure, prediction results (confusion matrix and accuracy) are provided in Tables 2, 3 and 4 below.

Table 2 Defined categories of Success Rates

Success Rate Category Label	Success Rate Range
A	100%
B	70 – 99%
C	50 – 69%
D	40 – 49%
E	0 – 39%

Table 3 Sample data structure for SVM learning

Total RAM GB	Total CPU cores	Application Memory Limit (%)	No. of Concurrent Users	Total Queries Run	Query Success Rate	Success Rate Category
976	128	34.02	5	45	82.22	B
976	128	34.02	8	58	55.17	C
976	128	34.02	8	68	29.41	E
976	128	34.02	10	70	40	D
976	128	34.02	12	110	10.91	E
976	128	34.02	12	106	6.6	E
976	128	34.02	15	135	5.93	E
976	128	34.02	15	62	14.52	E
1952	256	51.02	5	40	100	A
1952	256	51.02	5	37	100	A
1952	256	51.02	8	48	100	A
1952	256	51.02	10	30	100	A
1952	256	51.02	10	45	100	A
…						

Table 4 Confusion Matrix result of SVM prediction

	Prediction				
Actual	A	B	C	D	E
A	8	0	0	0	0
B	1	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	1	0	1

While the accuracy of the model is reasonably good (~72%), and able to predict the observations that belong to category A, the trained SVM model did not perform well on predicting other (less) success rate categories. This can be rectified by 1) fine tuning the SVM model; 2) continuous training of the model with new sets of benchmark data.

9 TUNING BEST PRACTICES IN BIG DATA AND CLOUD APPLICATION SCENARIOS

Big data Application deployed on cloud can be classified under complex deployment due to multiple controls and connectors. Controls can be security, monitoring etc. and connectors can be communication between multiple stack layers of the Big Data Platform.

Customer can see valuable insights on their data when the data pipelines starts form data ingestion → data storage in data lakes[2] → analytics → visualizations. At the same time all the controls and connectors must be working in optimum way.

Applications can be further classified into batch and real-time. Each of these scenario execution demands its own set of technology and tool stack and reducing processing time of batch jobs plays vital roles in every domain. for example, in ecommerce application which are global the competitive pricing can make or break the business. Hence, reducing processing time of batch jobs which calculates the pricing will allow to publish their prices faster and not lose the customers.

Similarly, for real time scenarios the server has to be always performant with in the SLA even if there is a surge in the traffic. This can be managed using Auto scale and Right scale feature.

[2] Datalake is a non-relational storage repository in Hadoop cluster that holds vast amount of raw data in their native format (such as log files, sensor data, images etc.

Table 5 Tuning Best Practices in various application domains

Layer	Tuning Best practice
ETL-Informatica	Tune Maximum Memory Allowed for Auto Memory Attributes to increase the job concurrency
Datalake-Hadoop	• By tweaking the # of reducer from default to standard formula helps 70 % reduction in processing time. • Hive OLAP, for complex queries, where computation work is more comparing to reading blocks it is recommended to use 128MB block size. Having more and smaller blocks helps to gain maximum parallelism and performance benefit.
Analytics-OpenR	It is recommended use data compression with RDS format for OpenR for better performance
Visualization-Tableau	Deploy Tableau desktop and server on different physical instances
Realtime streaming-Kafka- Spark	✓ Kafka partition should be equivalent to Kafka brokers ✓ To improve spark job concurrency further "spark.streaming.concurrentJobs" property needs to be set to higher value. Recommended value being a multiple of Kafka brokers

Table 6 below illustrates how we perform raw data analysis and draw out co-relation between running processes and resource utilization to be able to recommend required scale factor (SF) for the processes to run optimally.

Table 6 Raw data analysis and co-relation

No of processes	Memory Consumption per process (KB)	Total Memory Consumption (GB)	Inference
77	64056	4.70	Actual value: Average memory utilization per process is 0.2%
419	64056	25.60	Extrapolated values with SF at 80% of the available memory, system will be able to support only 419 processes [Note: System was unresponsive at 418 processes]
576	64056	35.19	Extrapolated values with SF for supporting higher processes
800	64056	48.87	Extrapolated values with SF for supporting higher processes

Finally, we provide some quick insights into workload patterns and system behavior under these workload patterns. Figure 3 provided below represents the impact to the SQL queries processing time with the increasing amount of work load (number of concurrent users issuing random queries of different complexities). The impact is measured in terms of scale factor on the processing time. This metric is beneficial in performing what-if simulation with varying workloads and patterns. For example, the queries classified as simple and represented in blue color have lower impact in terms of processing time whereas the queries classified as complex, have very high impact in the processing time when the number of concurrent users is increasing in the system.

Figure 4 provides the correlation between the increasing amount of workload (concurrent users) and the system resource utilization (%CPU, %RAM) at the Hadoop processes (such as Impala, YARN) and overall system level. The resource utilization is reaching the threshold limit with the increasing workload. This pattern helps in identifying the maximum workload that can be supported by the system at the infrastructure level.

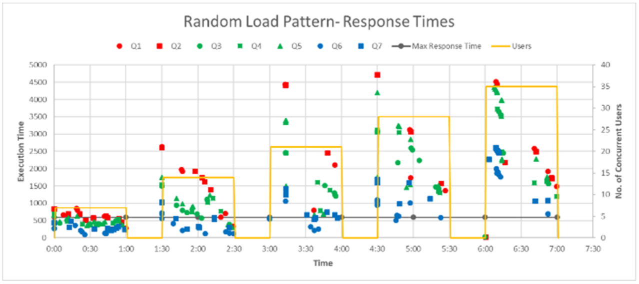

Figure 3 Random Load Pattern – Response Time

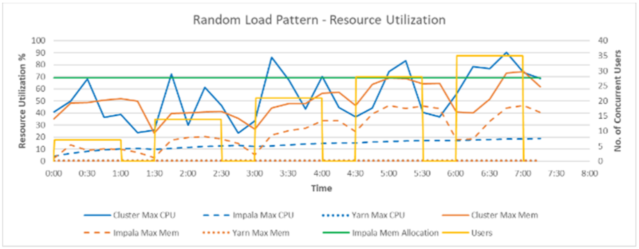

Figure 4 Random Load Pattern – Resource Utilization

CONCLUSION

In the paper, we presented the importance of adopting new AI techniques such as Machine Learning (ML) & predictive approach to performance benchmarking in a new paradigm of cost-effective Big Data and Cloud powered applications which is becoming a mainstream technology for enabling rapid deployments and delivering quicker insights in every application domain. We showed how it is going to be an improvement over the traditional approach to performance benchmarking to execute and measure performance. We started with laying out a foundation for lifecycle of performance benchmarking methodology applicable to all application domain with introducing a new concept viz. 'Right Scale' and its importance. We presented a high-level conceptual view of our automated performance benchmarking execution engine that we have designed and built to support a self-service automated benchmarking in every application domain.

To discuss the next level of our self-service model, we furthermore explored and presented two self-service use case examples to illustrate how the recommendation capabilities of our self-service automated engine can be powered by ML and predictive analytics models (derived from the rich collection of performance data) for making it more flexible in addressing complex user scenarios to predict 'Right Scale'. Finally, we also presented some real-world interesting industry examples with some tuning best practices, some real-world illustrations on correlating the raw output data from the tools with the system resource utilization to identify the hot spots.

ACKNOWLEDGMENTS

We want to acknowledge the support and encouragement we received from Accenture leadership in execution of the benchmarks in varied industry scenarios.

REFERENCES

[1] Demystifying Cloud Benchmarking Paradigm: An In-Depth View, Jul 2012, IEEE 36th International Conference on Computer Software and Applications COMPSAC 2012 [Jayanti Vemulapati, Venu Vedam]
[2] BigDataBench, A Big Data and AI Benchmark Suite, ICT, Chinese Academy of Sciences, http://prof.ict.ac.cn/
[3] YCSB, https://github.com/cloudius-systems/osv/wiki/Benchmarking-Cassandra-and-other-NoSQL-databases-with-YCSB
[4] TPC, http://www.tpc.org/information/about/about.asp
[5] TPC-DS, http://www.tpc.org/tpcds/
[6] Benchmarking – Wikipedia. http://en.wikipedia.org/wiki/Benchmarking
[7] Amazon Web Services. http://aws.amazon.com
[8] Amazon Instance Types. http://aws.amazon.com/ec2/instance-types/
[9] NetPerf. http://www.netperf.org
[10] NetSpec. http://www.ittc.ku.edu/netspec/