Future-ready IT Systems with Performance Prediction using Analytical Models

Madhu Tanikella

Infosys

Abstract

Large and complex distributed software systems can impact overall software cost and risk for enterprises, if they do not meet the performance & scalability SLAs when rolled out to production. The ability to accurately estimate the performance of such systems can mitigate this risk significantly. This can be achieved through Performance Modeling which can also act as the basis for Capacity Planning of IT systems in an organization. Such an approach helps plan effective procurement of hardware to serve the anticipated workload in future. Performance Modeling can be conducted using simple extrapolations, analytical models (simple mathematical formula-based and complex Queuing Network Modeling), and simulation techniques. However, analytical models represented by simple mathematical formulae are more preferred in the IT industry as they are cost-effective, while still being accurate making them more beneficial to IT organizations.

The objective of this paper is to introduce simpler, realistic and implementable analytical models based on the sound principles of Performance Engineering and Regression Techniques. Models described in this paper primarily revolve around simple and easy mathematical formulae and calibration using the available performance data points. Post-calibration, these models are ready for performance prediction of software systems at each individual layer (i.e., webservers, application servers and database servers in isolation), thus enabling organizations to be future-ready.

The role of Performance Modeling in Application Performance Management (APM)

The concept of Performance Modeling for distributed software systems has attracted considerable attention from organizations. The chief reason for this interest is the ability of this technique to enable ‘what-if’ analysis of a software system’s performance and anticipate change in user, transaction load, various software rollouts, or additional functionality (in terms of UI flows or modules). Gauging the impact of such changes on the performance of software systems well in advance holds the key to effectively planning investments in the areas of performance tuning and hardware procurement.

This helps enterprises support future workloads at the requisite quality-of-service (QoS) levels and thereby avoid any performance degradation.

Performance Modeling is a technique of predicting the performance characteristics of a software system such as end-to-end response time, throughput, or server utilization using various parameters such as load and concurrency with the help of models that require certain inputs and generate certain outputs.

Performance Modeling for a software system can be initiated at the design or architecture phase using minimal data points from a prototype (or proof-of-concept) and the models can be made more robust, accurate and effective with additional data points captured from subsequent build and test phases.

Performance predictions during design or architecture phases provide a basic idea about the system's performance characteristics. This understanding helps in making key decisions about the choice of software components and preliminary hardware requirements for test environments. Predicting performance in the design phase ensures early validation of system performance and scalability and addresses any architecture and design issues early in the software development life cycle (SDLC).This reduces the cost of identifying or remediating such issues in later phases, which otherwise can prove to be extremely expensive and difficult to resolve.

Factors crucial to the future performance of business-critical systems

When organizations invest in IT systems critical to business, the performance of these systems in varied environments with reaction to dynamic factors is of utmost importance to business stakeholders. While planning such investments, some of the most common queries that business stakeholders have for IT departments are:

- What are the response time and throughput characteristics of the application with current user load?
- What will happen to response time, throughput and server capacity if 'X' users or 'Y' transaction load is added to the current application?
- How does response time and throughput vary for different concurrent loads – what is the function of response time and throughput with varied concurrency?
- What is the maximum user load that can be supported by the existing hardware for this IT system while still providing acceptable performance?
- Will the existing hardware be able to support the anticipated user or transaction load in future?
- How do we ensure that the chosen design and architecture meet the required performance and scalability goals for the given concurrency requirements?

- Are there any simple and practical performance models that are cost-effective and easy-to-use with acceptable deviation?

In order to address these queries, many types of performance tests for load, stress and scalability are to be conducted on near-production-like hardware. This can be expensive and time-consuming in terms of procuring hardware and software licenses and executing multiple rounds of tests.

Performance Modeling can address the above in a cost-effective manner. Performance models can be built using performance data from a scaled-down environment. These models can then be used to predict the performance behavior of the system without actually executing all the tests that require expensive hardware infrastructure.

Performance Modeling using Analytical Models – the most effective technique for performance prediction

Performance Modeling can be conducted using multiple techniques such as simple linear extrapolation, analytical models (simple mathematical formula-based and complex Queuing Network Modeling), and simulation techniques. Fig.1 highlights the benefit of analytical models over other techniques which are at the center of the cost-accuracy spectrum.

However, analytical modeling can be further implemented either using simple mathematical formulae or using Queuing Network Theory / Layered Queuing Networks (LQN). The use of analytical models with simpler mathematical formulae is the preferred technique since these models are more cost-effective, easy to implement, and less demanding in terms of time and effort. The mathematical formulae used for this purpose are based on Performance Engineering principles (Little's Law [6], Scalability Law & Regression techniques [6]). Moreover, they generate prediction metrics with permissible deviation (around 10-20% of actual results) acting as a good reference point to start any capacity planning or hardware procurement effort for IT organizations. As the IT system is developed, these metrics are refined further using more data points for higher accuracy.

On the other hand, Analytical models using QNM and LQN techniques represent a given software system as a queue of resources (CPU, DISK, Memory, Network and Delay Centers) at each component level (Web/Application/Database Server) which better represent the system and hence the predictability and accuracy are higher. However, these techniques are slightly more time-consuming and complex.

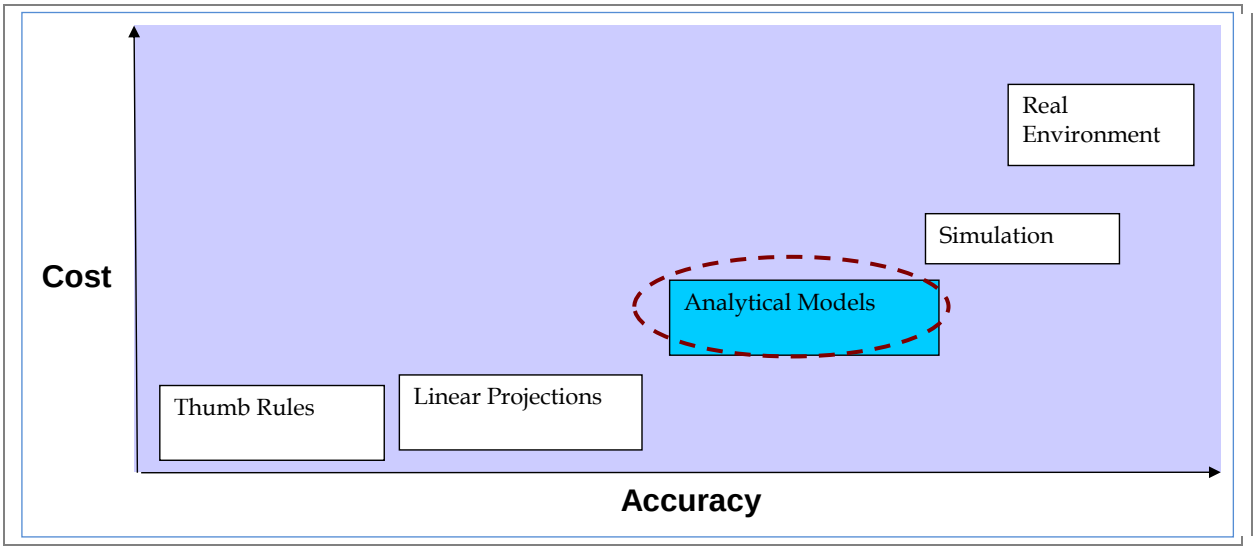

Figure.1. Performance Modeling techniques (Source: Author's representation)

Using analytical models for prediction

Let us discuss two analytical models that can be used to predict certain key performance metrics:

1. **MODEL - 1 to predict system throughput** for varying user load on a given hardware
2. **MODEL - 2 to predict maximum number of transactions** that can be supported by a given hardware

Model - 1: To predict system throughput for varying user load

The performance model presented below (Model - 1) can be used to predict throughput of a given software system on the existing hardware for future user load without performing tests for user volumes that cannot be simulated. This model also provides the break point or 'knee' of the given system (software and hardware) without conducting any kind of stress test.

Model-1 is represented by the following formula:

$$C(N) = \frac{N}{1 + \sigma [(N-1) + \lambda * N * (N-1)]} \quad \textbf{[1,2,3]}$$

where

C(N) = Relative Throughput = $\frac{X(N)}{X(1)}$

X(N) = Actual Throughput for N users

N = Number of Users ***(INPUT)***

σ = Level of Contention for a resource in Hardware, Software or both that represents the percentage of Serial Execution ***(specific to a given System)*** *[1,3]*

λ = Level of Coherency (accounts for the time taken to fetch a cache miss in CPU cache etc.) ***(specific to a given System)*** *[1,3]*

Model Inputs: **User Load**

Model Parameters: σ & λ (to be calibrated)

Data required for calibration: **load and throughput**

Model Output: **throughput**

Re-writing the equation and comparing with **y = ax2+bx+c** (a parabola)

$y = \sigma \lambda x2 + \sigma (1+ \lambda) x$ ----------> [Equation 1]

Where $y = \frac{N}{C(N)} - 1$ and $x = (N-1)$

Comparing Equation 1 with y = ax2+bx+c

$a = \sigma \lambda$ ----------> [Equation 2]

$b = \sigma (1 + \lambda)$ ----------> [Equation 3]

Solving Equations 2 and 3 for σ and λ,

$\sigma = (b-a)$ ----------> [Equation 4]

$\lambda = \frac{a}{(b-a)}$ ----------> [Equation 5]

Calculating model parameters

In order to use the above model, the *'Model Parameters (σ, λ')* should be calculated and calibrated for a given system. For this, a minimal set of data is required. For instance, if performance prediction needs to be conducted for 500 concurrent user loads, performance metrics for 10, 25, 50, and 70 concurrent users must be captured in a scaled-down environment, model parameters must be calculated, and then the throughput for 500 user loads can be predicted using Model - 1.

(Note: The data points mentioned above are merely to illustrate the point and they may vary in each case).

Once the throughput for various user loads is captured, the data can be regressed with $y = ax2+bx+c$ and values for 'a' and 'b' can be obtained. Using these values, the model parameter values **σ** and **λ** can be obtained (using Equations 4 and 5).

Predicting metrics using Model - 1

To illustrate Model - 1, Standard Performance Evaluation Corporation (SPEC) Software Development Environment Throughput (SDET) benchmark data [4] is used. Table.1 below shows predicted throughput, calculated using Model - 1. This value is close to the actual throughput captured from SPEC SDET benchmark (% deviation is in the range of 10-20%).

(Note: Only the first four data points highlighted from SPEC SDET data are used to create the model and calibrate model parameters.)

SPEC SDET Benchmark Data[4]**		**Data calculated using Model-1**	
Concurrent Users	**Actual Throughput X(N) (scripts per hour)**	**Predicted Throughput X(N) (scripts per hour)**	**% Deviation**
1	64.9	64.9	0%
18	995.9	889.16	11%
36	1652.4	1377.19	17%
72	1853.2	1785.3	4%
108	1828.9	1871.31	2%
144	1775	1839.24	4%
216	1702.2	1673.94	2%

Table.1: Throughput comparison using Model-1 (Source: SPEC SDET Benchmark [4] and author's work)

Model - 2: To predict the maximum number of transactions that can be supported

The performance model explained in this section (Model - 2) can be used to predict the following parameters without actually performing a stress test:

- The maximum number of worker processes or threads that can be supported by a given hardware
- CPU utilization of the system for a desired user load

Model inputs: Required/threshold CPU utilization **Model parameters**: System characteristics such as contention, caching, etc. **Data required for calibration:** Active threads and CPU utilization ***Model output: Active threads for given CPU usage***

This model is based on the fact that the CPU utilization on a given server is directly proportional to the number of active worker threads or processes running at a given point in time, and activity performed by an end-user is executed by a thread or process on any given server. As the number of active worker threads or processes increases with the increase in concurrent user load, the CPU utilization increases and eventually reaches 100% beyond which the CPU utilization remains at 100%. However, the response time starts degrading since requests queue up waiting for the CPU to be available. Figure 2 below represents this relationship between active threads and CPU utilization on a given server.

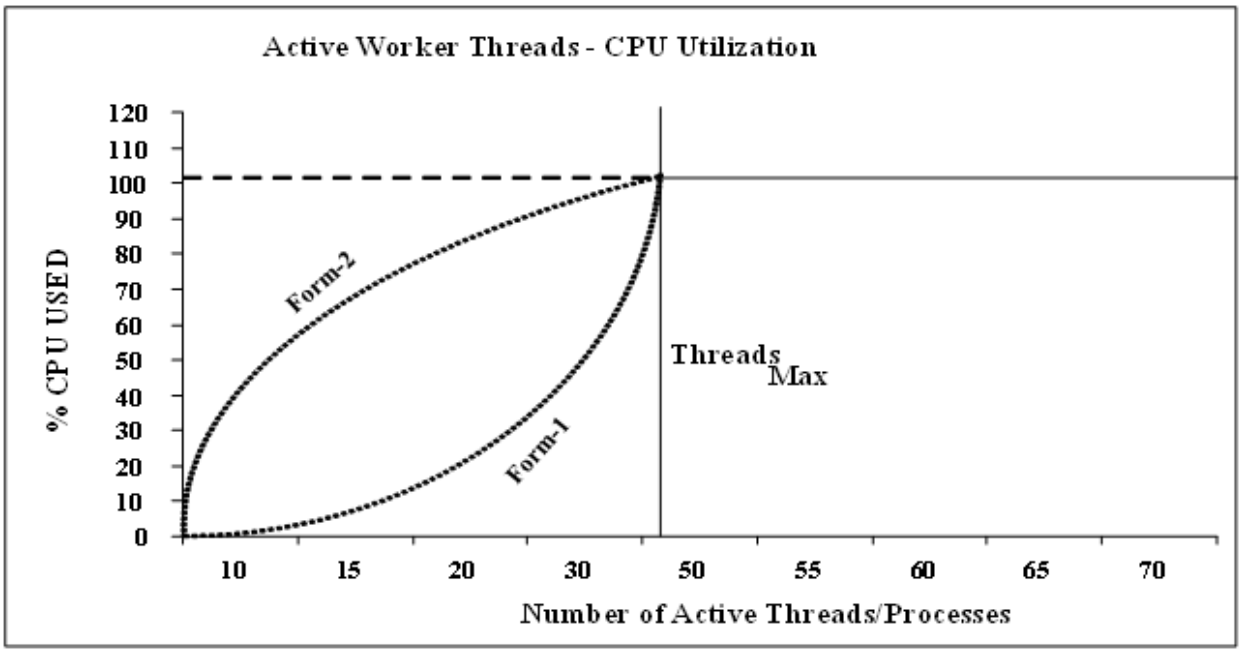

Figure.2. Relation between active threads and CPU utilization (Source: Author's representation)

From Figure 2, within the region marked by **Threads$_{max}$**, the relationship between active threads and CPU utilization follows either Form-1 or Form-2 depending upon the nature of the application and usage pattern, which could follow any non-linear function such as log, power, exponential. The choice of the non-linear curve depends on the best-fit criteria for the performance data collected for a given system.

Model-2 is best applicable to web servers, application servers and any other type of server where the amount of work done by a worker thread or process is similar in nature and hence a direct correlation can be drawn between the server threads or processes and server CPU utilization. It is assumed that any downstream software component in the architecture (for example, database, EIS, SAP, PeopleSoft) is properly tuned and sized in such a way that the wait times at these components do not affect the processing of the software component to be performance modeled.

To demonstrate the use of Model - 2, CPU utilization and active threads data is captured from an application server in test environment for different user loads from a working project. A Scatter Chart for these data points is created using MS Excel and a trend line is chosen based on the best fit that is represented by the correlation coefficient (R) **[5].** The closer the value of |R| to 1, the better the fit to the given data points, thereby making it a fairly accurate mathematical model.

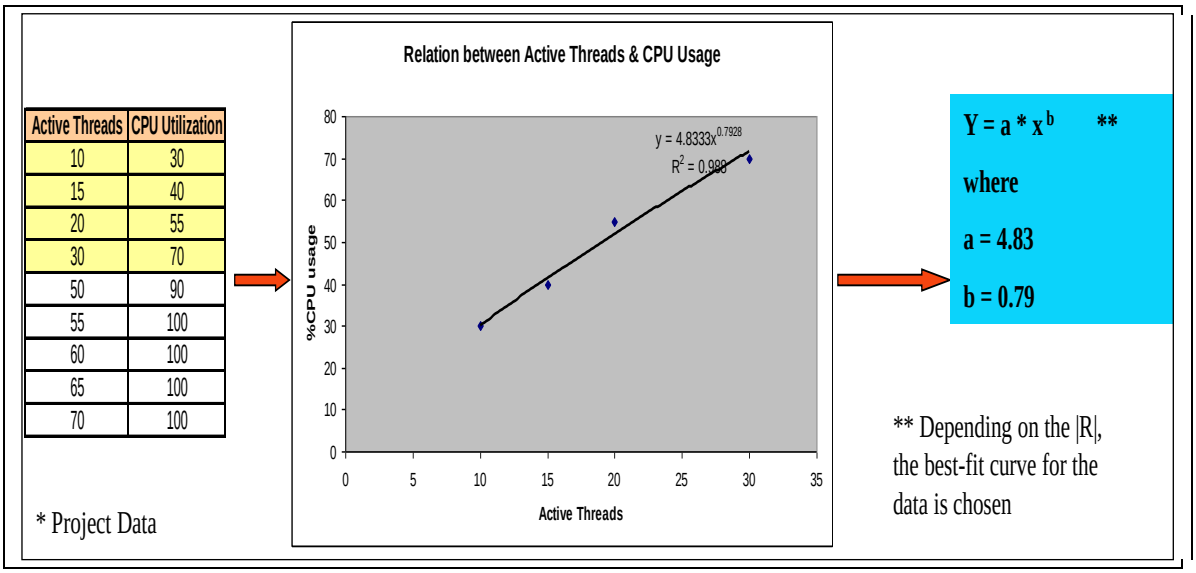

Figure.3. Relation between active threads and CPU utilization (Source: Author's Representation)

Figure 3 represents the analytical model that has been created using only four data points from the active threads/CPU utilization table and it is represented by a power function, $\mathbf{y = a*x^b}$, where the model parameters ***a*** and ***b*** represent the characteristics of

the given system such as contention for resources, inherent service delay, and queuing at different resources.

Model-2 is represented as $\mathbf{y = 4.83 * x^{0.792}}$, where a=4.83 and b=0.79, which in turn is used to predict the following:

- For 100% CPU utilization (y = 100%), x ~= 46. This represents $\mathbf{Threads_{max}}$ in the system that results in 100% CPU usage. This represents the maximum user load supported by the given system on the given hardware.
- From the actual data, y = 100% occurred at x = 55. This implies a 16% deviation in the predicted and actual active thread.
- The CPU usage can be predicted for any active threads using the model,

By capturing user load and throughput data points for this system along with active threads and CPU usage using Model - 1, the maximum concurrent user load that can be supported by the existing hardware can be calculated. This can be used to map the active threads and concurrent user load.

Also, by using Little's Law **[6]**, the average response time for the maximum user load can be predicted for the given hardware:

Concurrent users in system = Throughput * average response time

Performance Prediction – Other techniques & considerations

Performance Modeling is one of the most complex disciplines; it can be implemented using different techniques such as Analytical Models and Simulation Techniques.

Another technique that can be used for Performance Modeling will be using *Software Simulation.* Although this can be more accurate, it is complex to build and implement. Some simulation software products are available as COTS (commercial-off-the-shelf) products. These are licensed, so prediction accuracy comes at a cost and hence the adoption is slower by the organizations

Besides, each IT system is unique in terms of its architecture, usage pattern, and complexity, making Performance Modeling unique for each system. Performance Modeling needs to be built from design phase and made robust and refined as the IT system evolves. Predictions are more accurate for systems that are tuned for performance and scalability.

Conclusion

The analytical performance models discussed in this paper represent light-weight mathematical models which are easy to implement. These models predict performance metrics of enterprise software systems, namely throughput, maximum concurrent user load, response time, and CPU utilization for anticipated loads in future with acceptable deviation and minimal data collected from a scaled-down environment. These models can reduce the effort required to run several incremental load tests and also help save on licensing costs of load simulation tools. The validation of these models using data from real-world enterprise applications is currently in-progress.

References

1. Gunther, N. J., The Practical Performance Analyst, iUniverse.com Inc. 2000.
2. Gunther, N. J., Lecture notes for Guerrilla Capacity Planning, Performance in the Valley Series of instructional classes for 2000
   http://www.teamquest.com/resources/gunther/display/12/index.htm
3. http://www.perfdynamics.com/Test/gcaprules.html
4. http://www.spec.org/osg/sdm91/results/res9506/ - SPEC SDET benchmark data
5. Statistics for Managers Using Microsoft EXCEL, Levine, D., Berenson, M., Stephan, D., New Jersey: Prentice-Hall (1999).
6. Little's Law: http://en.wikipedia.org/wiki/Little's_law