ITARC 2009

Web application performance analysis

Rajesh Chellamani
Principal Architect,
CT, a Wolters Kluwer Business

Agenda

- Introduction
- Performance analysis scenarios
- Prerequisites
- Queuing Theory Introduction
- Queuing theory application
- Application architecture evaluation
- Challenges
- Conclusion
- Reference

ITARC 2009

INTRODUCTION

Introduction

- What is system performance analysis?
  - Response time
  - Resource utilization

Performance analysis scenarios

- Production system experiencing performance issues
- Performance analysis of application architecture
  - Identify potential performance issues early in application development process
- Capacity planning
  - What is the maximum load level supported by the current server configuration?
- Server consolidation and hardware specification determination

ITARC 2009

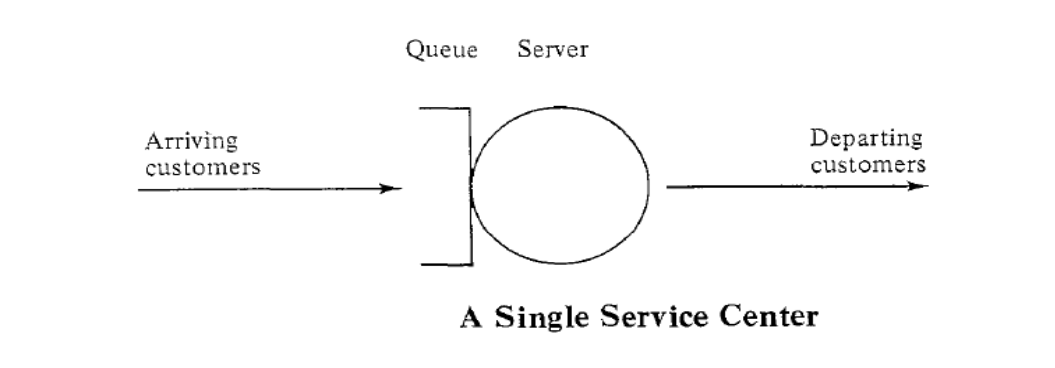
QUEUING THEORY

Basics of queuing theory

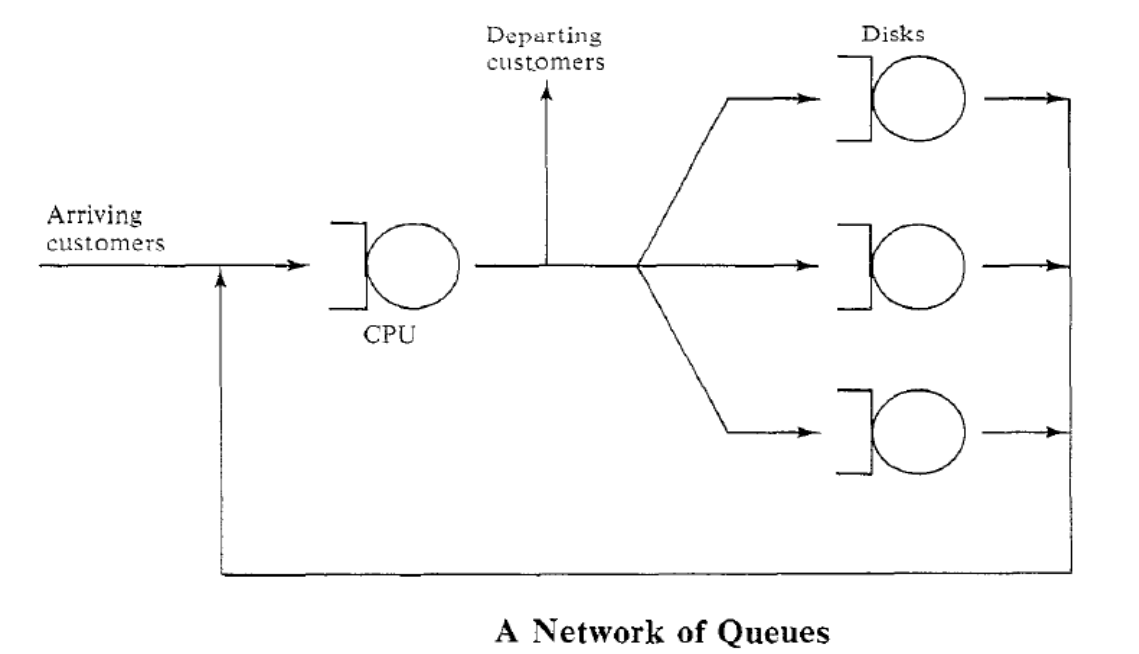
Queuing Theory - Introduction

- Simple method to analyze ways to analyze system performance using mathematical models
- Very generic – applicable in many analysis situations that involve “customers” and “service centers”
- Naming convention: uses Kendall Notation. Contains 5 parts
  - Input arrival type: Probabilistic (memory-less), General …
  - Service time: Probabilistic (memory-less), General …
  - Number of service points: 1, m, …
  - Queue depth: 1, m …
  - Algorithm: FIFO, LIFO …
- Example: M/M/1, M/M/m

Example of Queues



A Single Service Center



A Network of Queues

ITARC 2009

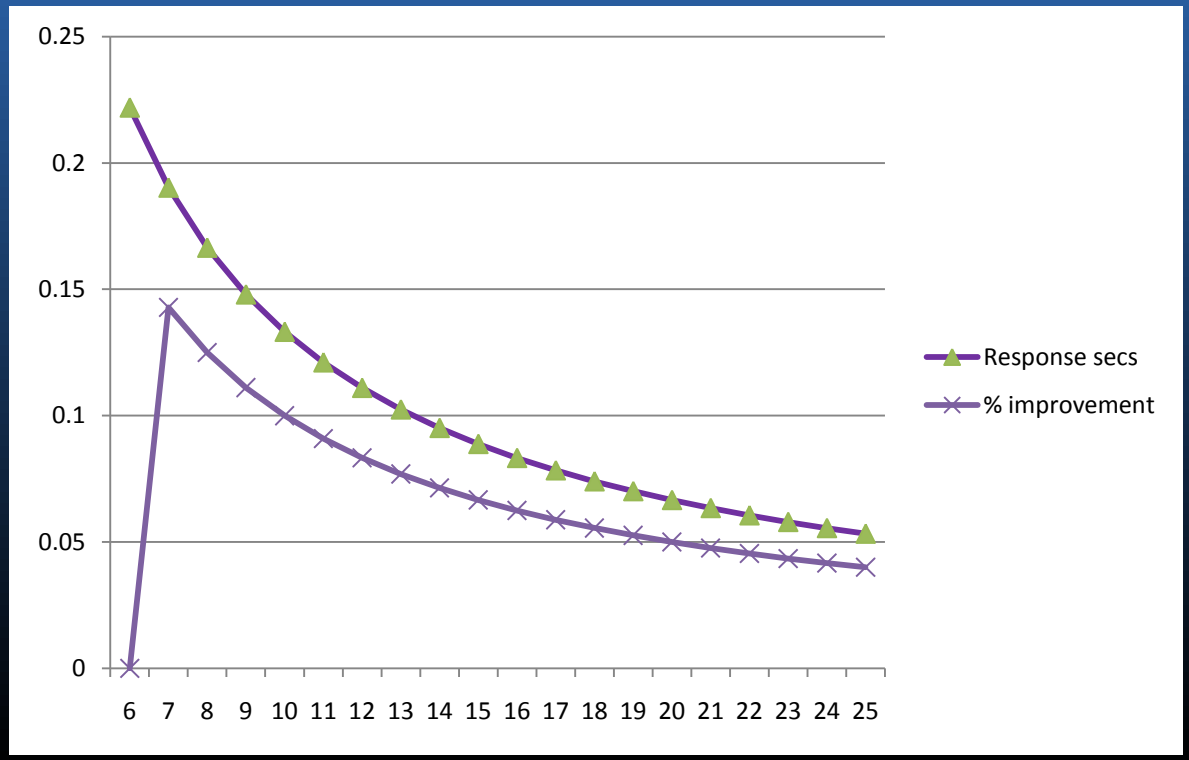
QUEUING THEORY

Application Performance Analysis

Database resource utilization model

- High CPU usage in production
- Decision: Hardware upgrade
  - Add more CPU – add 2 more or 4 more CPUs?
  - Add more memory
- Process
- Challenges

Modeling database performance



Model creation process

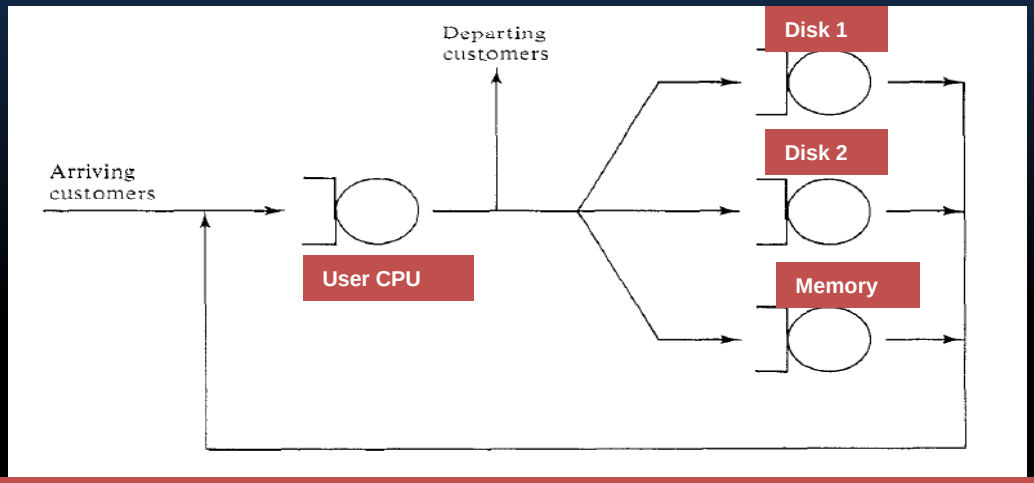
- Define the objectives
- Identify the system and queues
- Identify the parameters
- Define/Select an existing model
  - Iteration 1: Choose existing M/M/m model
  - Iteration 2: Create a new model
- Validate the model
- Predict and repeat measurement and validation

Steps in custom model creation

- Step1: Analyze request pattern, average response time and DB processor utilization of the current system
- Step2: Workload characterization for the DB server
- Step3: Document the system configuration, tools used and assumptions in the model
- Step4: Create a simple model by analyzing the current system
  - DB Response time = Time spent processing request + (Disk IO time + Memory IO time)

Custom model creation

- Objective: Determine the CPU usage in the database server after hardware (HW) upgrade
- Analyze server logs to confirm that Oracle process is the most active
  - Eliminate any external factors influencing performance.
  - User CPU + System (kernel mode) CPU time represents the request processing time



Queuing model to study impact of adding processor and memory to database server

Definition of model parameters

- t: time after hardware upgrade
- t-1: time before hardware upgrade
- Cu(t-1): User CPU at time (t-1)
- Cu(t): User CPU utilization at time t
- Cs(t-1): system CPU at time (t-1)
- p(t): Number of CPUs at time t,
- p(t-1): Number of CPUs at time (t-1)
- M(t): Memory at time t
- M(t-1): Memory at time t-1

Definition (cont)

- δ: Fraction of additional CPU at time t
- α: data fetch efficiency. Assumed a constant of 0.15 (i.e system cpu at time t-1)
- γ: Additional CPU available due to adding more memory

$$\delta = 1 + \frac{p(t)}{p(t-1)}$$

$$\gamma = 1 - \left[\frac{M(t) - M(t-1)}{M(t-1)}\right] * \alpha$$

Custom model: conclusion

$$Cu(t) = \delta(1 - Cs(t-1) * \gamma)$$

- More CPU available for processing request
- As per this model, if γ = 0.15, δ = 1.66 and C_s(t-1) = 15%, C_u(t) will improve by 38.37%
- Observed improvement after hardware upgrade was 60% improvement
- Additional 20% attributed to redesigning a resource intensive application model

Application performance evaluation- Preparation

- Instrumentation framework
- Requires historic data for database and web server performance
- Application system dependency
- Steady state performance profile
  - Database
  - Web services and applications
  - NAS/SAN storage
- Document typical performance numbers for application components

Application architecture evaluation (cont)

- Waterfall SDLC process
- Produce existing usage profile of existing applications
- Define application performance requirements and goals
  - Acceptable response time
  - Acceptable resource utilization
- Document typical response time based on existing implementation
- Assign performance quotas
- Adjust for errors/unknowns

Application architecture evaluation (cont)

- Create a component inventory
  - Analyze and document component dependency tree
- Performance quota allocation
- Identify potential performance bottlenecks
- Redesign if required and repeat the process

CONCLUSION

Challenges

- Business justification
- Frequent upgrade to applications changes response time and resource utilization profile
- Infrastructure changes
- Other factors
  - Performance testing
  - Environment differences

Conclusion

- Keep the model simple
- Re-evaluate the model periodically
- Performance testing – certify applications on a hardware setup of lower capacity than the production servers
- Define performance buffers

References

- Guerilla Capacity Planning – Neil J. Gunther, Springer Publications, www.perfdynamics.com
- A short introduction to queuing theory – Andreas Willig, Technical University Berlin, Network Telecommunication Group, http://www.tkn.tu-berlin.de/curricula/ws0203/ue-kn/qt.pdf
- Quantitative System Performance: Computer System Analysis Using Queueing Network Models*:* Edward Lazowska, John Zahorjan, G. Scott Graham, Kenneth C. Sevcik (http://www.cs.washington.edu/homes/lazowska/qsp/)
- Developing and Applying a Distributed Systems Performance Approach for a Web Platform – James Cusick and Terry Welch, C T Corporation, {James.cusick, Terry.welch}@wolterskluwer.com

Questions?