Thomas Clark

5/4/09 cs162 lecture notes

cs162-aw

Performance Evaluation

Recommended reading: Heidelberg and Lavenberg "Computer Performance Evaluation" IEEETC, C33, 12, Dec. 1984, p. 1195

We've been talking about the concepts of operating systems, this lecture is about measuring real systems and making them faster and more efficient

Performance modeling and measurement is needed through the entire life cycle of a system: design, debugging, installation, data collection for the next system

Performance evaluation covers the following areas: measurement, analytic modeling, simulation modeling, tuning, design improvement

Good work in performance evaluation requires a good understanding of the system under study. You can't just arrive with a bag of tricks (e.g. queuing theory) and do something useful

Need to plan for optimization throughout design process, developers cannot "just create, then optimize"

Measurement:

- Measuring gives advantage o having real results with real workloads
- Measures all of the interactions that have to take place, which can escape mathematical models
- Disadvantage is that it takes time and effort to get data
- Not that many published measurement studies

Hardware Monitoring: to get some real numbers

- Use some sort of hardware monitor (e.g. logical analyzer) to collect and partially reduce data
- Disadvantage is that hardware monitoring is slow
- Uses probes clipped onto the circuits to record data (in old days, there were discrete transistors to clip on to)
- Hard to use hardware monitors today because we often want to monitor a piece of logic in a chip, but there are only 3 chips on a computer
- Hardware monitors can count events, get samples, see when things arrive
- Can only trace for a short time, if we only have MBs or GBs of data with a GHz processor
- Have to be sure that sampling times are correct, we don't want samples to be skewed so that we never see certain interactions happen, can sample at random times to solve this problem

Software Measurements:

- Can set up interrupts
- Some systems have built-in profiling tools, but they slow the system down
- Can use microcode to collect data

- Can cause 20% or more overhead

Modern CPUs have counters built-in to count instructions, tlb misses, cache misses. Intel has a tool to set a few registers to get this data.

Multics Measurement:

- Has timer interrupts
- Hardware counter for memory cycles
- Had external I/O channel to be externally driven to read sections of memory
- Remote terminal emulator to drive system

Diamond:

- Internal DEC measurement tool – a hybrid monitor. Hardware probes read the Program Counter (PC), CPU mode, channel and I/O device activity

IBM GTF (general trace facility):

- Debugging tool
- All kinds of interrupts and traps
- Trace records on interrupts and traps, I/O interrupts, start I/O's, opens
- Problem is that everything is designed for debugging, but this produces a lot of data, much of which is useless
- Another problem is that the traces in a system do not always present data usefully. To find out which process accessed which word address won't be in the same trace record, even though they are related in concept.

IBM SMF (system management facility): generates records for assorted events – jobs, tasks, opens, closes

Can use the SMF and GTF together for results

Some mainframes have console monitors which generate real time info and measurements

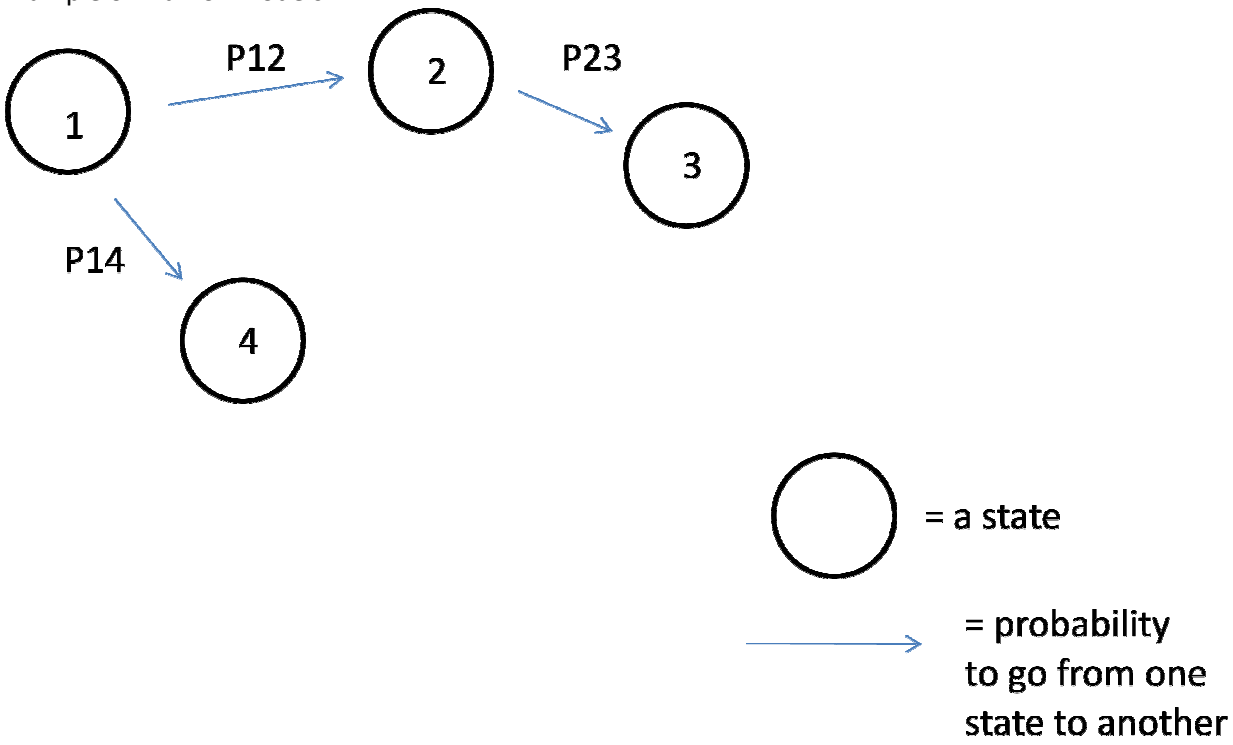
Workload characterization:

- Important for any kind of workload study
- 3 types of workloads for performance experiments
    - Line workloads – good for measurement but not repeatable (e.g. a guy typing on a computer)
    - Executable – real samples, always same workload, repeatable
    - Synthetic executable workloads – parameterized versions of real workloads, may not be representative of reality

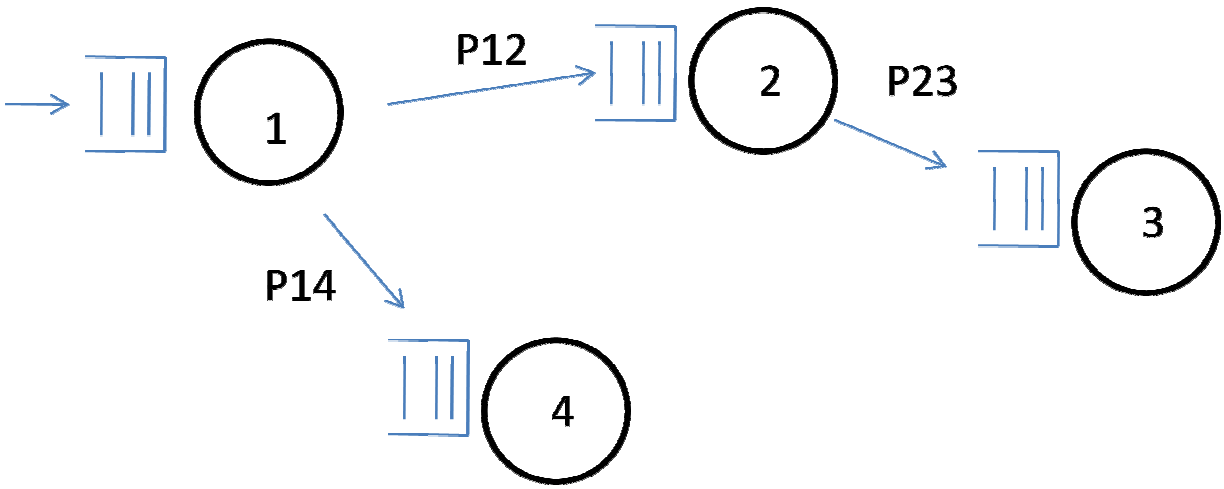
Designing a workload model:

- How workloads act on future systems is generally related to how past systems run, because the future and past systems are generally related
- To characterize a workload we have to decide what is relevant (many research papers are not useful because the workload models are not accurate of reality)

- To figure out how to characterize workloads we have to analyze real data, can use exploratory data analysis and cluster analysis
- Statistical methods to find accurate workload models
    - Means, variances, distributions
    - Techniques like linear regression, factor analysis
    - Can use statistical analysis on data to see if various models fit
    - Seldom used – little intersection between class of competent experimental performance analysis and competent statisticians
- Analytic Performance Modeling
    - Build analytic model of some type of system of interest
    - Example of markov models



- Calculate factors of interest as function of parameters
- Lots of research, some of it is useful
- Models to be queuing models stochastic process models
- Most of progress in queuing theory has been made in the last 30-40 years due to computer system modeling
- Can also use BCMP models for I/O, continuous random processes, an example below



- Advances in modeling give us a class of queuing theory for network models which are easily solved, efficient computational algorithms for these solutions, and good approximation methods for systems that are not easily solved

Pros/Cons of Analytic Modeling:

- Good for capacity planning, I/O subsystem modeling, preliminary design aid
- Capacity planning is a big area of application
- Does not have fine detail, probabilities good enough for us

BCMP:

- Customers have type T
- Routing system can be of form p(I, t1, j, t2) where I,j are servers and t1, t2 are types
- Servers can be FCFS exponential, with a service rate a function of queue length

Simulation:

- Discrete event simulation – trace driven, random number (stochastic)
- Continuous simulation (e.g. a differential equation), not used for computer system modeling
- Monte Carlo methods (sampling) – if we had a 2-dimensional blob and wanted its area we could make a square around the blob and randomly choose points to test and see if they're in the blob
- Simulation models have a model of the system, which has a state. And a set of events which changes the state. And a method for generating such a sequence of events, and a measurement component, which records the statistics of interest.
- Discrete event simulation is the general simulation model
    - Events come from event list
    - Update system state
    - Event list updated
    - Statistics accumulated
- Several languages for simulation (GPSS, simscript, gasp, slam, simula)
- Simulation modeling packages, resq

Analysis on simulation output:

- Regenerative simulation – find regeneration point, do standard IID statistics
- Time series analysis – do time series analysis – also called spectral method
- Repeat entire simulation run
- Very long runs – hope that results converge
- Batch runs – take long samples, and consider them IID

Keep in mind that back of the envelope calculations can get pretty good results, and only take a few minutes to solve.

An example of a back of the envelope calculation is to approximate the number of gallons of water that flow through the Mississippi River in a year. We can guess the cross sectional area and speed of the water. We can also guess that the Mississippi River collects the water between the Rocky and Appalachian Mountains, and can guess that half will be lost to evaporation, and guess the amount of rainfall. Both of these methods can get results in the same order of magnitude, without requiring a complex model.