Lecture 3
Capacity Planning Process

Models

Parameters

Workload Characterization

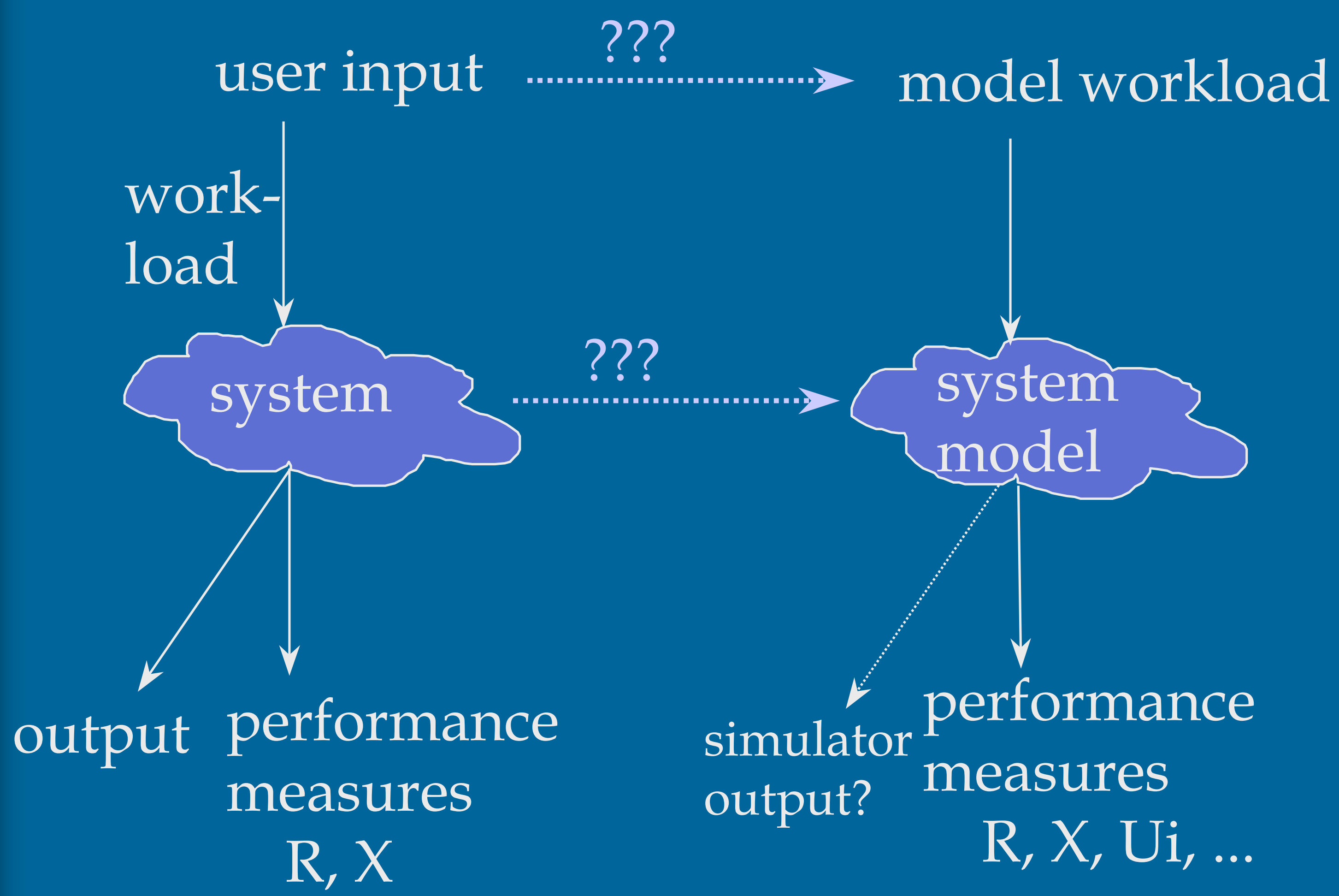
Clustering

Workload Forecasting

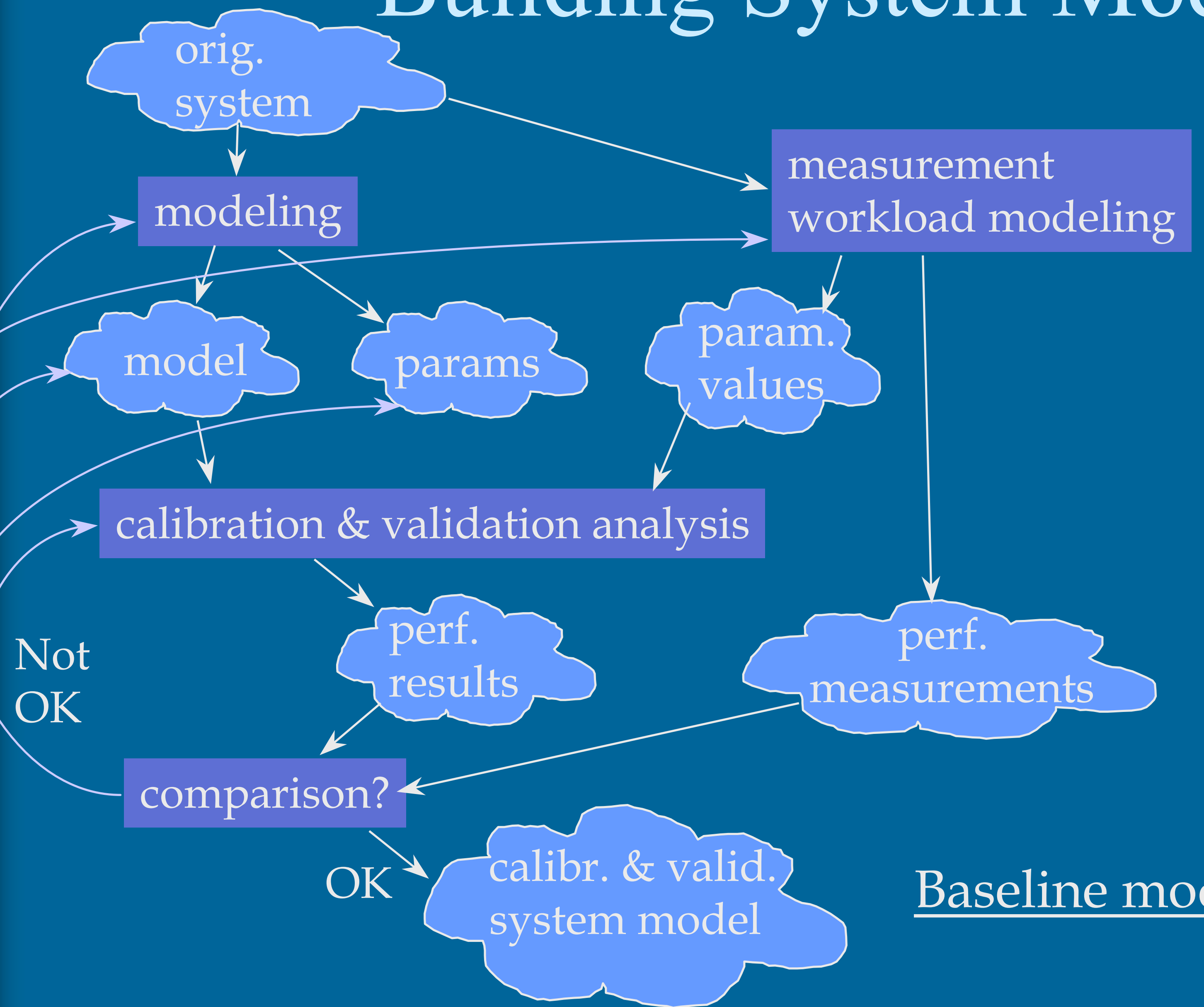
Capacity Planning Process

- How is it used?
  - fig. 2.1 [Men 94]
- Overall methodology
  - fig. 2.3 [Men 94]

System Model

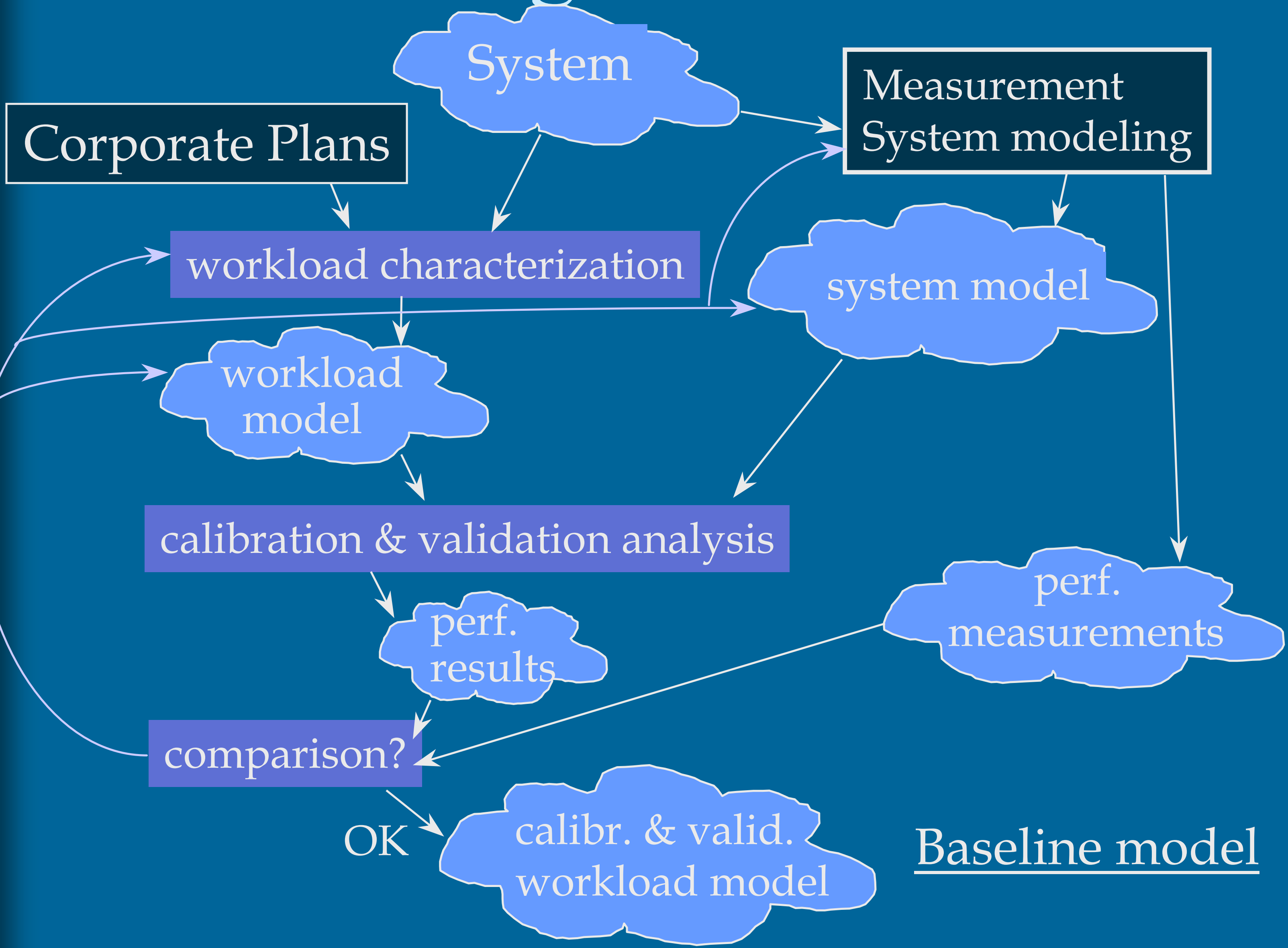


Building System Model



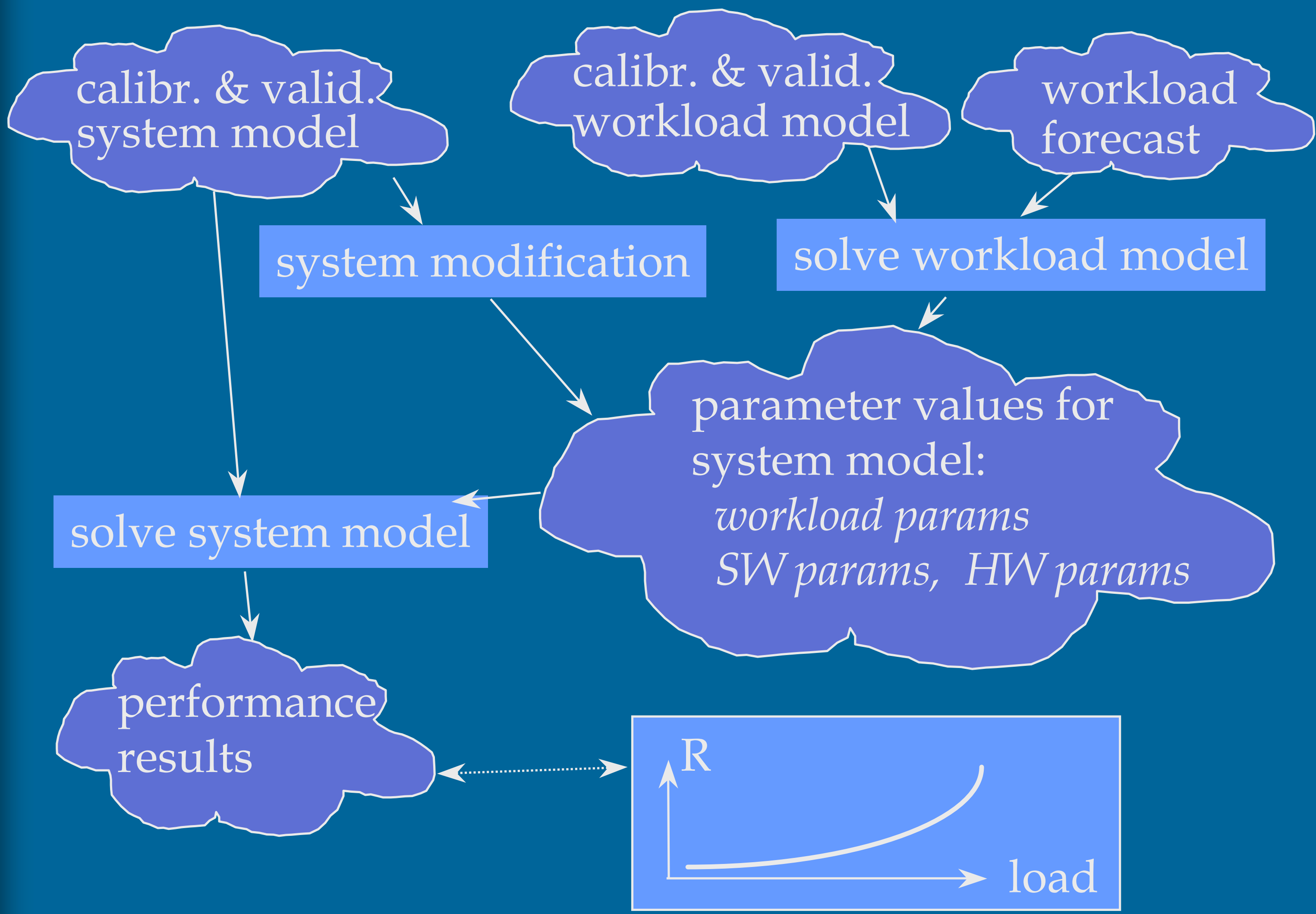
Baseline model

Building Workload Model



<u>Baseline model</u>

Use Models in Forecast



Estimating Parameters (5)

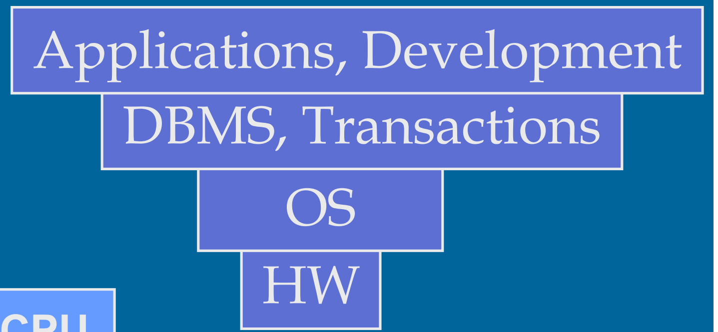
- CPU time for one Query?
  - usually routines A, B and C invoked
  - total nr of instructions executed in average: 2000
  - architecture speed: 2.3 clock cycles per instr.
  - 4600 clock cycles, 10 MHz cycle time, 4600/10000 ms = 0.46 ms
- Faster CPU? 15 MHz => 0.31 ms
- Better compiler? 1000 instructions => 0.23 ms
- New disk driver? new architecture?
- What if big variance in nr of instructions executed?
  - new job class? more details....

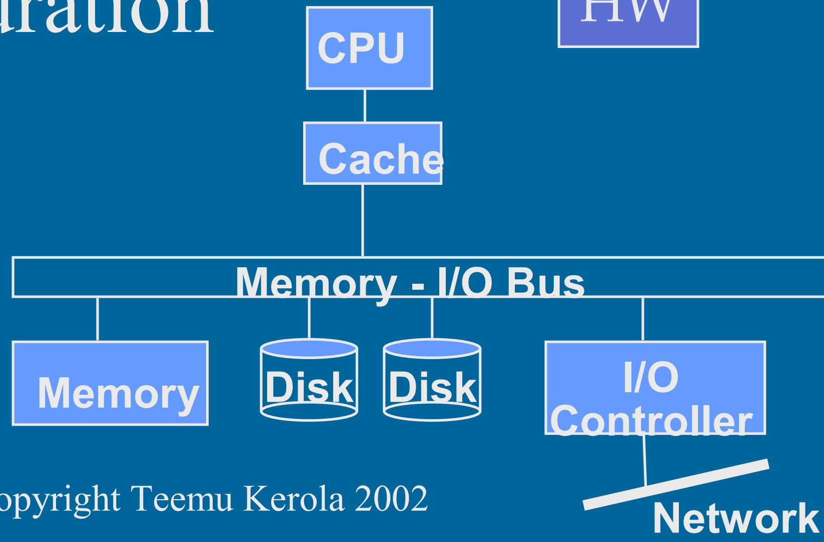
Estimating Parameters (7)

- CPU time for one Query?
- generate only Queries for 1 hr
  - disk completely utilized, no good data
- generate 10000 queries at 50 ms intervals
  - disk no a problem now?
- average CPU utilization 10.2%, time 80 s,
- CPU util when no Queries present: 4.5%
  - network, other work?
- Util for Queries: 5.7%, cpu time 5.7% x 80 s = 4.56 s
- cpu time per Query: 4.56 s/ 10000 = 0.46 ms

What is Hard in Modeling?

- Need to understand whole system and environment before modeling
- Computing environment
  - resources, capacities
  - service requirements
- Device configuration
- Own SW
- Workload





More Details in Model?

- Get more useful information
- System model becomes more complex
- Have more parameters; need to estimate them all
- Workload model becomes more complex
- Model solution becomes more complex
- E.g., if system load is high and transactions must queue for free memory partitions, then one should include memory in the model

Less Details

- Might not get enough useful information
- May leave out something really important
- E.g., disk transaction time may depend on
  - I/O channels, queues for heads of SCSI strings
  - device controllers, disk caches,
  - file access protocol, rotation speed
  - etc etc
- May use average service demand $D_{class,disk}$ if not interested in where the time is spent

Performance Prediction Methods

- Rules of Thumb
- Trend Analysis
- Operational Analysis
- Performance Models
  - analytical, simulation
- Benchmarks
  - best one is your own

Job Classes or Transaction Classes

- Workload may have different classes of transactions (jobs)
- Complicates model and parameter estimation
- Complicates solutions and use
- Gives more usable information
- Can aggregate some classes to simplify model

Priorities

- Job classes may have priorities at some resources
  – CPU
- Priorities may be dynamically changing
- Difficult to model well
- Complicates solutions

Capacity

- What is capacity?
- Theoretical capacity
  - fig. 2.5 [Men 94]
  - pop N = Resp. Time * Throughput
  - E.g.,

$$0.6 = 0.2\,\frac{\text{sec}}{\textit{transaction}} \times 5\,\frac{\textit{transactions}}{\text{sec}}$$

open system?
flow out (X, throughput)
= flow in

- Effective Capacity
  - when some defined constraint is reached

How to Compare Different Systems to Each Other?

- Time, throughput
- Benchmarks
  - standard (Whetstone, Linpack, …)
  - own
- MIPS
- SPECmark
- TPC - Transaction Processing Council

When is Load Measured?

- Peak load
  - fig. 2.7 [Men 94]

What do you Measure?

- Performance
  - CPU utilization
- Other
  - availability
  - reliability
  - cost

Requirements

- Something that must be met, or system fails
  - response time (e.g., less than 1 sec.)
  - throughput (e.g., at least 650 tps)
- MTTF - Mean Time To Fail
  - fig. 2.8 [Men94]
- Availability
  - fig. 2.8

$$MTTF = \frac{48+72}{2}$$

$$Availability = \frac{161}{168} = 95.8\%$$

Performance Goals

- Throughput $X > 30$ tps
- Response time $R < 2$ sec
  for 90% of the transactions
- Aver. resp. time for trivial transactions
  - $R_{triv} < 1$ sec
  - $T_{CPU} < 5$ msec, max 5 I/O's

Performance Goals (contd)

- Availability > 98% (monthly?)
- command response time
  - local commands: < 0.5 sec
  - remote commands: < 1.5 sec
- MTTF > 10000 hours
- What system?
  - PC? Bank? Nuclear Power Plan? Airport control system?
  - real time? hard real time?

How to Characterize Workload?

- Example: mail Teemu.Kerola < aa.txt

12 times

- Basic components: 12 jobs
- Functional characteristics
  - software components?
  - mail? name server? file server? network server?
- Resource characteristics
  - CPU time? I/O time? (per disk?)
  - memory usage?

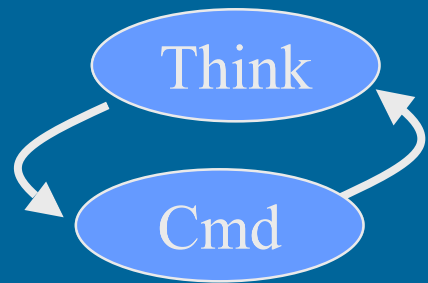
Workload Characterization

- Input, output:
  - Fig 2.10
- Representativeness
  - Fig. 2.11

Mail Example

- 12 mail commands
- Resource usage
  – Tbl 2.1 [Men 94]
- Simple (single class) characterization
  – Tbl 2.2
- More complex characterization
  – cluster jobs into 2 classes based on response time
  – Tbl 2.3
- Input rate (λ)? Think time (Z)?
  – Fig 2.12

Which one is better?

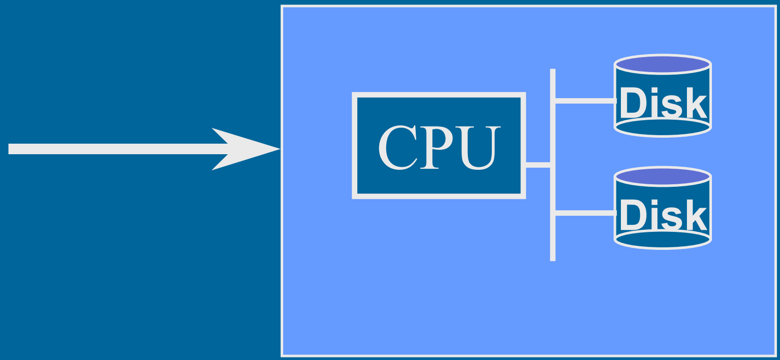


Workloads

(koetinkuormat)

- Real
- Synthetic
  - use part of real workload?
- Artificial
  - executable, not executable?
- Hybrid
- Kernel
- Non-executable
  - some abstraction of real workload
  - input for models
  - e.g., *average* service demand

File Server Load



- Request distribution?
  - arrival rate? interarrival times? per class?

10 trans/sec

- File access probabilities?

60% disk-1

- File access file size distribution?

10% at most
1 sector, ...

- Read/write distribution?

10% are writes

Workload Partitioning

- Based on some attribute
  - resource usage, application,
  - geographic orientation, functional attribute
  - etc etc etc
- Tbls 2.4 - 2.10 [Men 94]

Clustering

- How to determine that one should use multiple classes?
- How to determine what jobs belong to which class?
- Cluster analysis (statistics)
  - fig. 2.13 [Men 94]

Clustering Metrics

- How to decide if two jobs belong to the same class?
  - Tbl 2.11 [Men 94]
- Euclidean metric

$$d_{i,j} = \sqrt{\sum_{n=1}^{K} (D_{in} - D_{jn})^2}$$

  - geographical distance between two jobs (i, j) based on K attributes (n)
  - see p. 54 [Men 94] Euclidean measures
  - used scale may affect result - beware!
  - Does B&C make up a class? or A&B?

Normalized Euclidean Metric

– Z score for each job *i* for each attribute n

$$Z_{in} = \frac{D_{in} - \overline{D}_n}{std.dev(D_{jn})}$$

for attribute n over all jobs j

– Distance measured as Euclidean distance between (normalized) Z scores (instead of raw measurements)

– See measures on p. 55 [Men 94]

– Another method to "normalize": use logarithm of the values instead of orig values

  • good when magnitude varies a lot

Clustering Algorithm

- k-means algorithm
  - Fig. 2.15 [Men 94]
  - Example, Tbl. 2.12-18 [Men 94]

Workload Forecasting

- What is the value of some model parameter 5 months from now? 1 year from now?
- Must have this info to use models for forecasting

System Model Parameters

- Workload model
  - use workload estimate to obtain system model parameter estimates
- Workload
  - Key Value Indicator
  - Natural Business Unit
  - Forecasting Business Unit
- System Model Parameters
  - Data base request arrival rate & distribution
  - Think time for interactive users

Forecasting Techniques

- Trends
  - Fig. 2.16 [Men 94]
- Moving Average
  - Tbl. 2.21
- Exponential Smoothing
  - feedback loop from most recent forecast error

$$f_{t+1} = f_t + \alpha (y_t - f_t)$$

  - Tbl. 2.22, Fig. 2.17

Forecasting Techniques (contd)

- Regression Models
  - estimate is dependent on other variables
    - estimated variable (e.g., y) difficult to measure directly
    - dependent variables (e.g., x) easy to measure

$y = \mathrm{a} + \mathrm{b}x$

  - make a small set of measurements for dependent variables (x) and estimate (y)
  - select factors (e.g., a & b) so that errors in estimate are smallest
  - use same factors for new dep. variable values
  - method of least squares
    - Tbl 2.23 [Men 94]

Performance Prediction

- Output: forecasts for different scenarios
  - Fig. 2.18 [Men 94]
- Different techniques
  - low vs. high complexity and cost
  - Fig. 2.19
  - Fig. 2.20