New Challenges for Performance Engineers:

Analyzing On-line Services

Varsha Apte

Associate Prof., IIT-Bombay.

(*Part of the work was jointly done with: Mohit Gupta, TCS)*

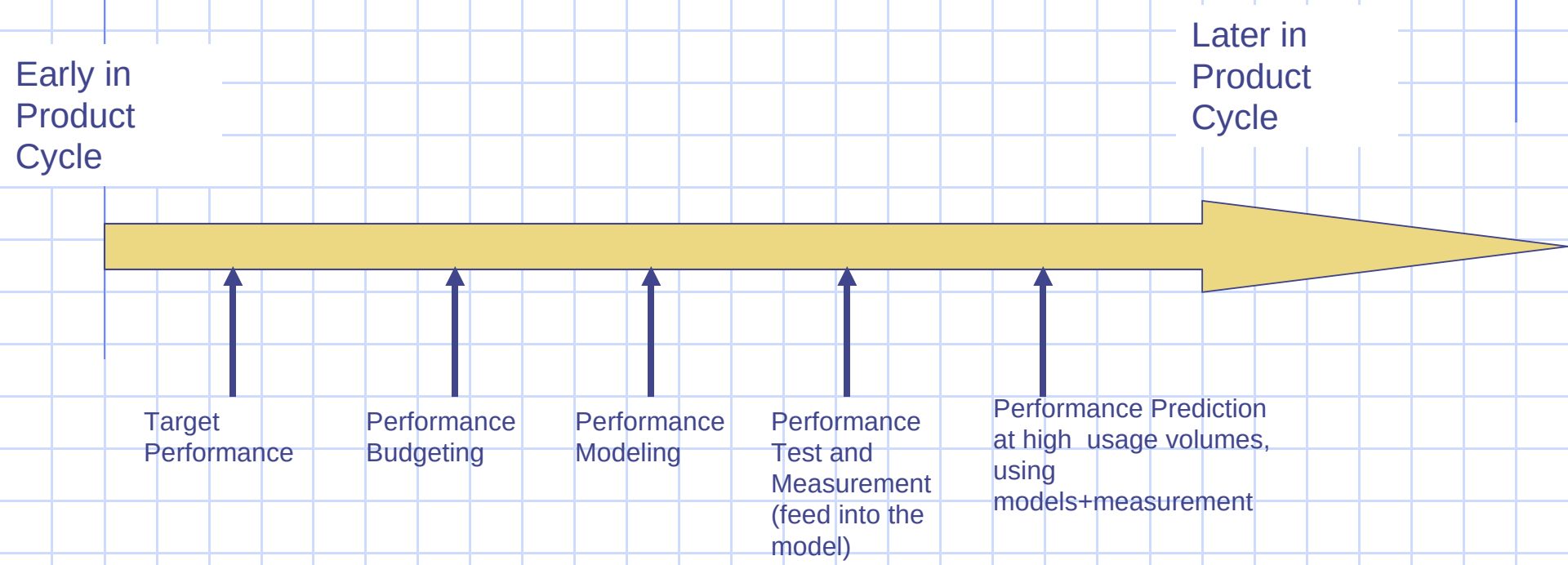
Outline

- Motivation
  - Product performance engineering techniques are ineffective when designing services
- Performance of on-line services
  - Challenges
  - Existing approaches
  - Ongoing work at IIT-B

Context

- A global shift is happening towards a “service economy”, often enabled by the Internet
  - Many technology providers are shifting focus towards *services* or *systems integration*
- Pressure towards accelerating time-to-market of services
- Has impacted how performance evaluation is done

Elements of Performance Engineering - Product



e.g. Internet routers, switches, Web-servers, Web back-end software, application servers, DB servers

Elements of Performance Engineering - Product

Early in Product Cycle

Later in Product Cycle

- Performance Requirements
- Performance Budgeting
- Performance Modeling
- Performance Test and Measurement (feed into the model)
- Performance Prediction at future usage volumes, using models+ measurement

- **Performance modeling involves modeling internal details of product**
- **Detailed models can be used in choosing product design**

Measurement analysis is tightly coupled and verified with analytical models and with developers

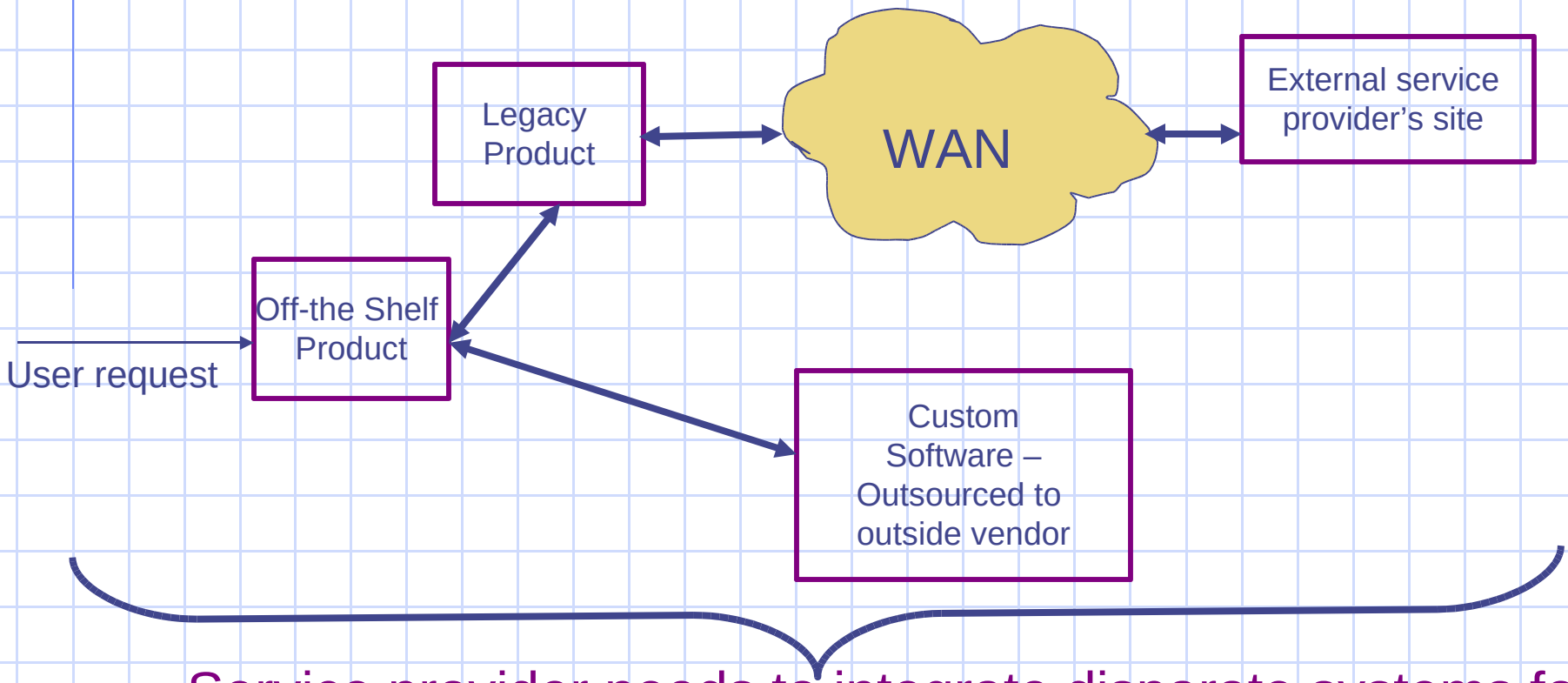
Tight coupling with development team

Enter: Services

- E-commerce web-sites
  - Banking
  - Shopping
- Web-based e-mail service
- Technical support service

Service Architecture-

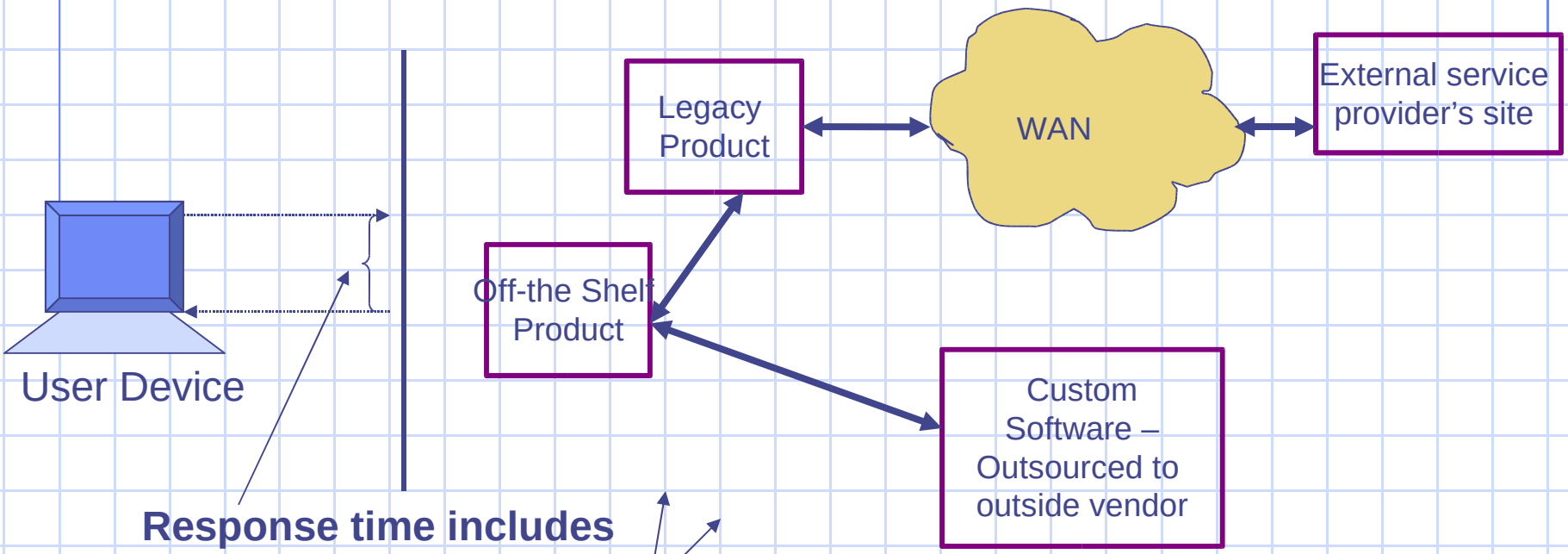
- Typical web-based service



Service provider needs to integrate disparate systems for providing a composite, seamless service

Service Performance Engineering –

Assuring good user-perceived performance



Challenges in Performance Engineering of Web-services

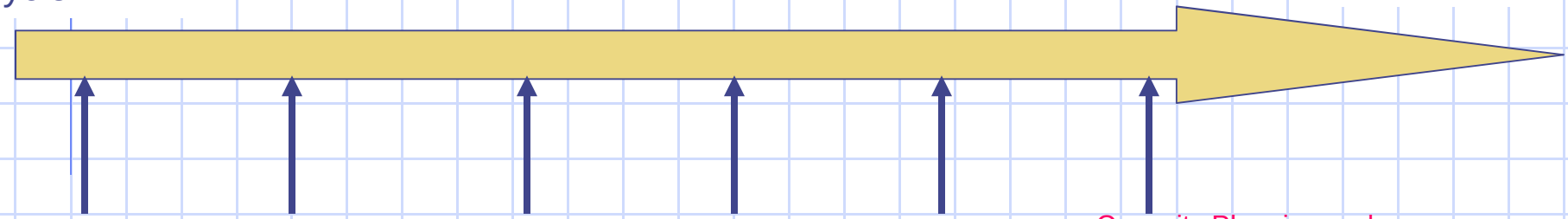
- Internal details of products may not be known
  - Internals of off-the-shelf products are protected as IP
  - Custom software developers may be
    - Geographically “far away”
    - Not very eager to share details
  - Not much may be known about legacy systems
- No control over external systems

Elements of Performance Engineering - Service

What are the differences?

Early in Service Cycle

Later in Service Cycle



Workload Character-ization/ Forecast Gathering

Performance Requirements –end-to-end

Performance Budgeting only on boxes over which there is some control

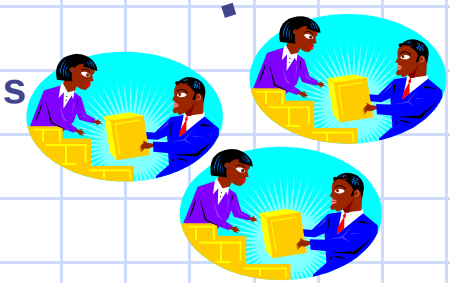
Performance Modeling of end-to-end delay, of public-domain protocols

Performance Test and Measurement at "black box level" (feed into the model)

Capacity Planning and Sizing of Service Centers for future usage volumes, using models+measurement

Weak coupling. Performance analysts use information but cannot give feedback

Development teams



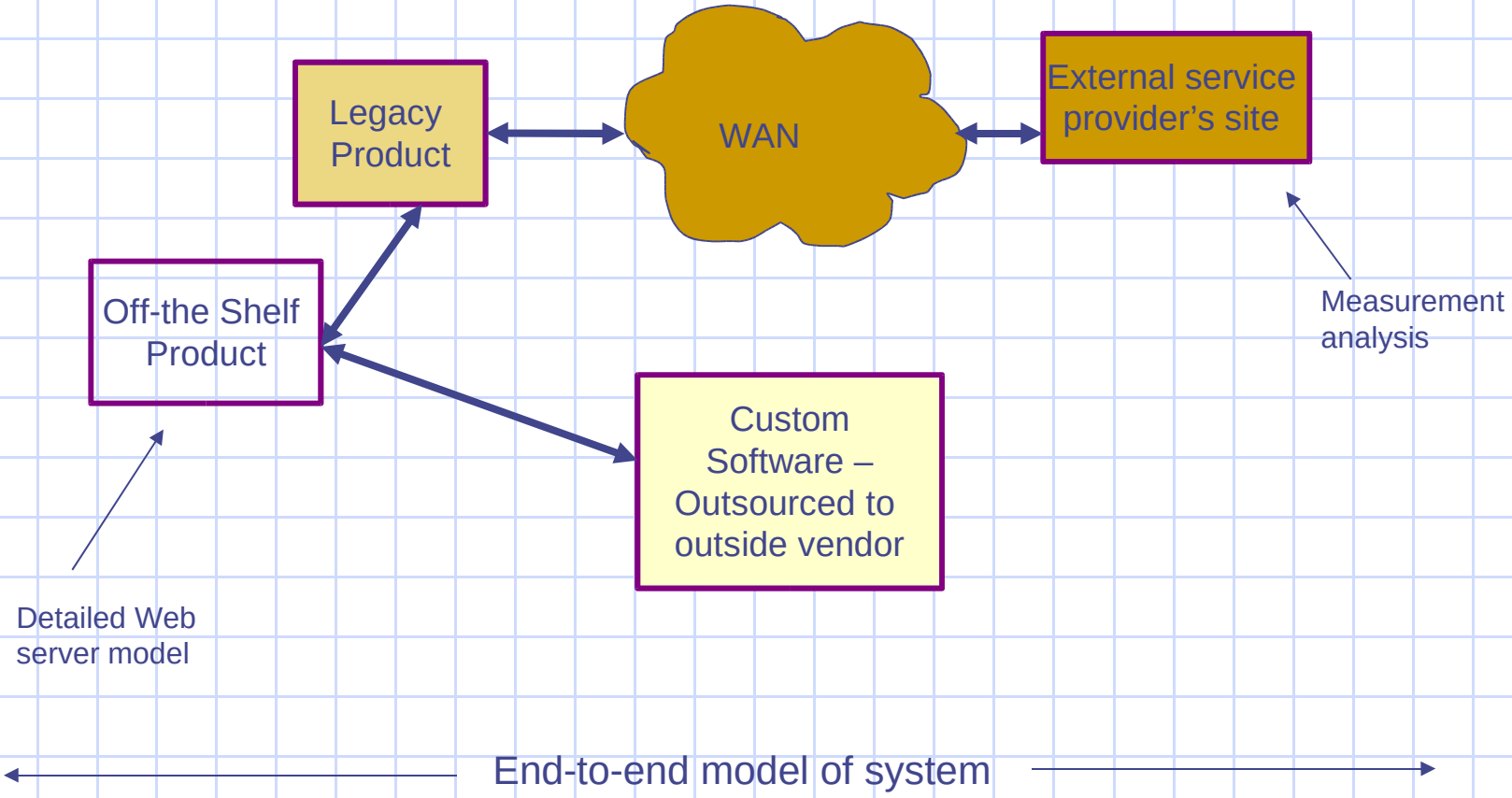
So What's the Point?

- Advanced queueing models for performance analysis not possible/not useful
- Focus has to shift instead to the means available and the needed information

“Means” and “Needs”

- Means:
  - Measurement analysis of black/“gray” boxes
  - Simple models for high-level architecture
  - Detailed models of well-known technologies (e.g. Web-servers, TCP/IP, SSL))
- Needs
  - Capacity analysis, sizing analysis, bottleneck analysis

… “Means”


Legacy Product
WAN
External service provider’s site
Off-the Shelf Product
Custom Software – Outsourced to outside vendor
Measurement analysis
Detailed Web server model
End-to-end model of system

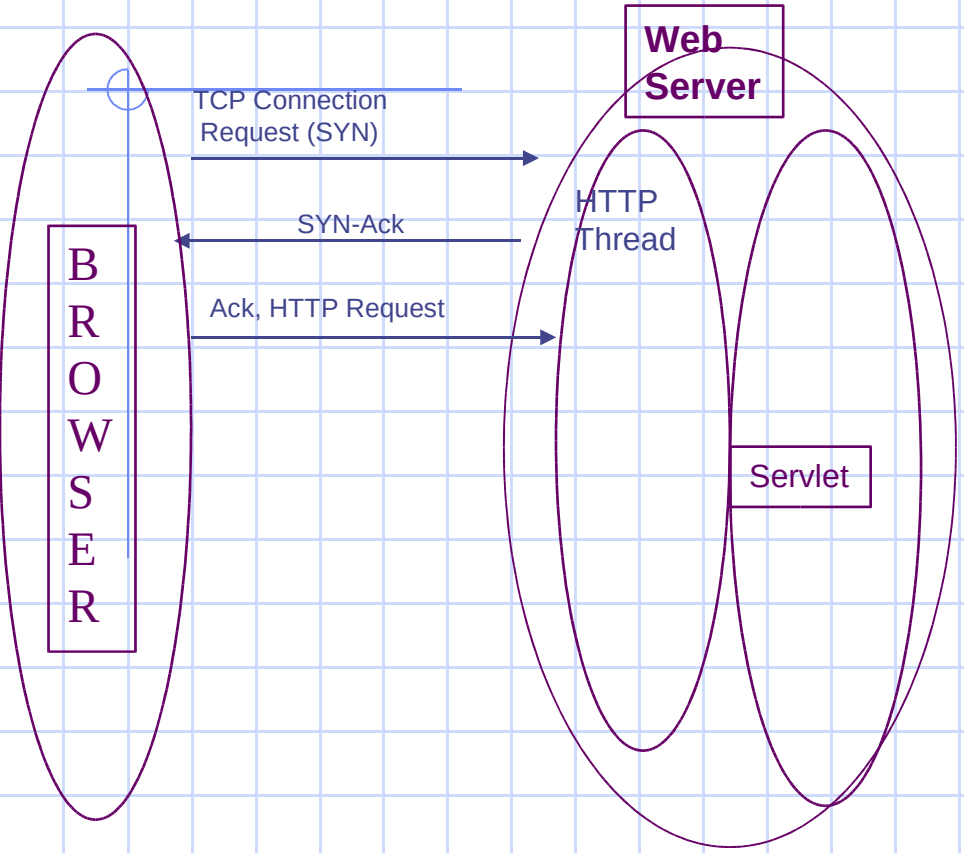
Analysis Approaches

- We'll discuss these three "means"
  1. Models of well-known technologies, in this case, Web-server
  2. Measurement-based analysis
  3. End-to-end modeling of systems

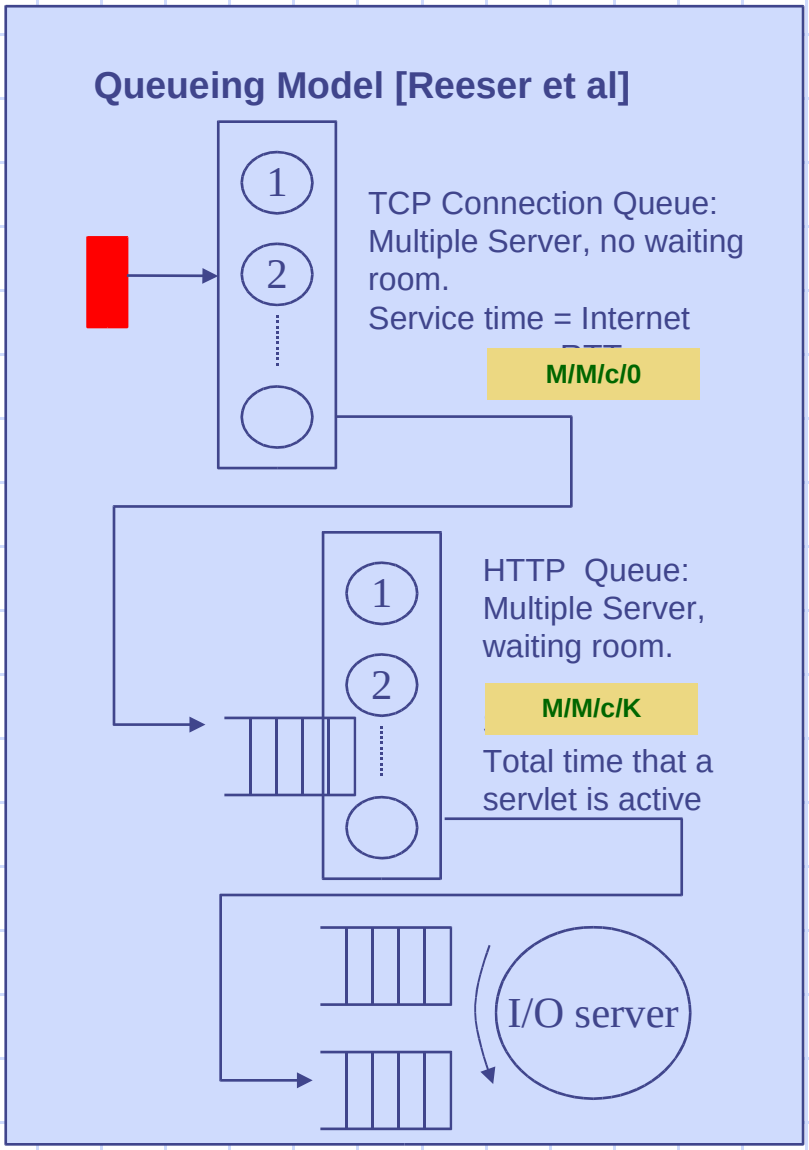
1. Web-server Models

- Various queueing models proposed
  - Reeser et al [1] first proposed a detailed model which captured all aspects of a Web server which serves static files
  - Mainkar [6] as well as Reeser et al [2] extended this model to represent dynamic Web-servers

Web Server Queueing Model



Servlet = Thread spawned by a Netscape-type Web server, to handle dynamic processing
RTT = Round Trip Time



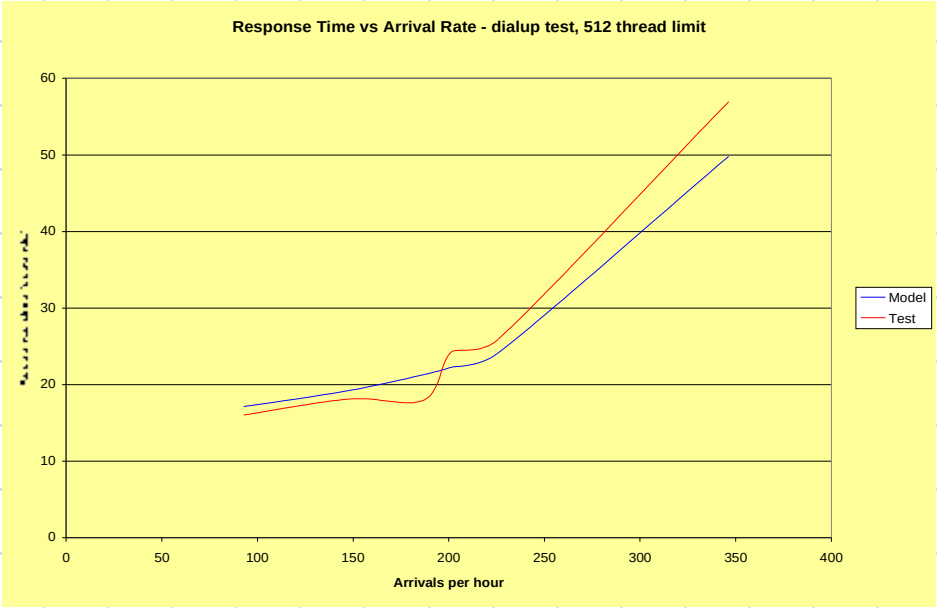
Web Server Queueing Model

- Original queueing model captures details of system I/O queues and the rate at which they are “drained”
  - Shows that web-server throughput depends on whether users access it mainly over dial-up or over a LAN (lower when dial-up)
  - Has deep impact on how results based on performance measurement on a LAN are extrapolated to a dial-up scenario

Dynamic Web Model

- Two layered model (requests queue at HTTP threads, HTTP threads queue at CPU)
- Solved using iteration

- Dynamic server model validated with *tests*
- Validation shows good results



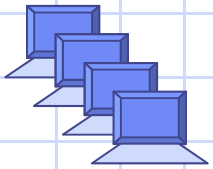
2. Performance Measurement

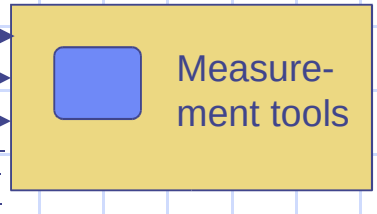
- For web-based services,
  - Off-the-shelf load generator and performance monitoring products
- Performance measurement may have to be of a “black box” (internals not known)

Performance Measurement – *Tools*

- Commercial load generator tools focus on ease-of-use for “system test group”
- There is a need for better tools targeted towards performance analysts

Testing team focus: Check if service meets requirements



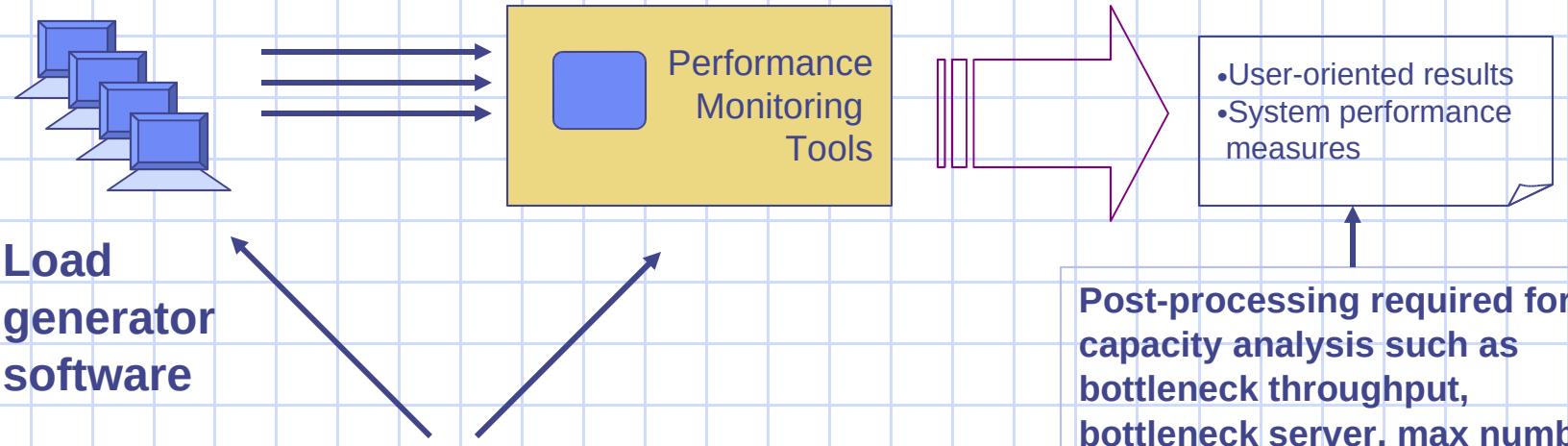


Load generator

Performance analyst’s focus: Take everything into account and produce a performance/capacity analysis, sizing plan, as well as architectural improvements

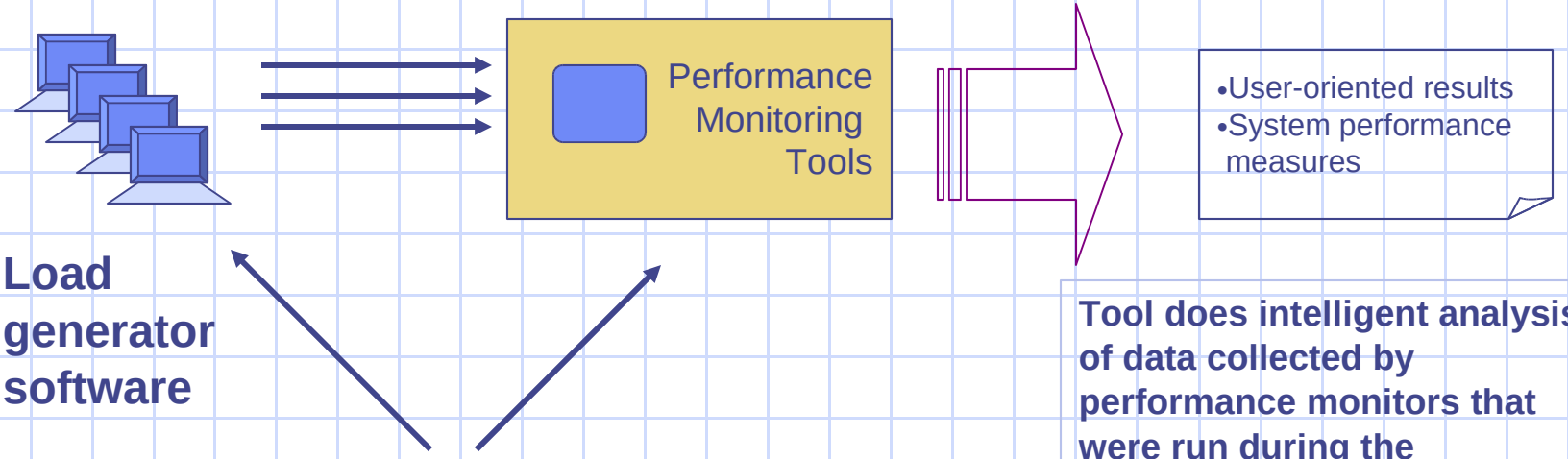
Performance Measurement – *Tools*

- Using Existing Tools for Capacity Analysis:



Performance Measurement – *Tools*

- Ongoing work at IIT-B (nascent stage)

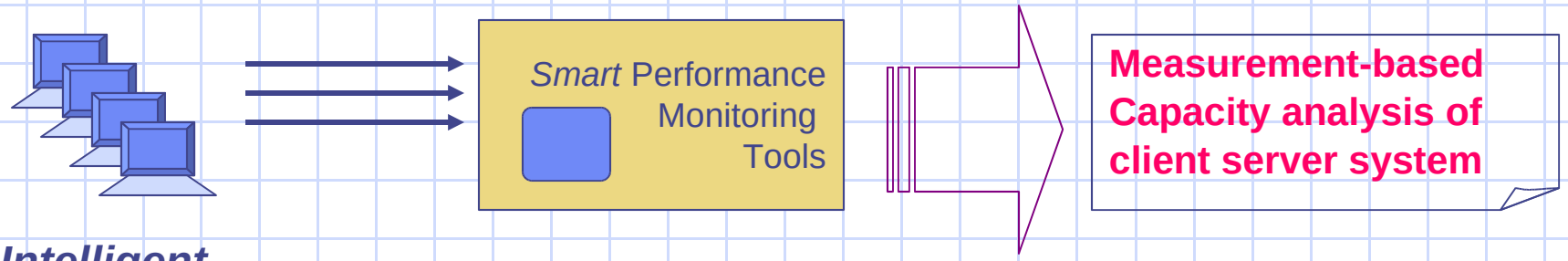


Tool does intelligent analysis of data collected by performance monitors that were run during the measurement period.

Tool that intelligently co-ordinates working of load generator and gathering of performance statistics at the server (e.g. rules for detecting steady state, for range of load over which measurement is to be done)

Performance Measurement – *Tools*

In short, the tool's aim is:



Intelligent **Load generator software**

Performance Measurement – *New Challenges*

- "Box" internals are not known
- Apart from capacity analysis, diagnosis of performance problems may be required
- Analyst can work only with measures collected by operating system

Performance Measurement – *New Challenges*

- Different approach required for such analysis
- *Signature-based* analysis is one such approach, described in [3]
  - *Signatures* are characteristic, repeatable behaviors of server software
  - Approach involves deducing the performance problem by observing *measurement signatures*

Signatures example

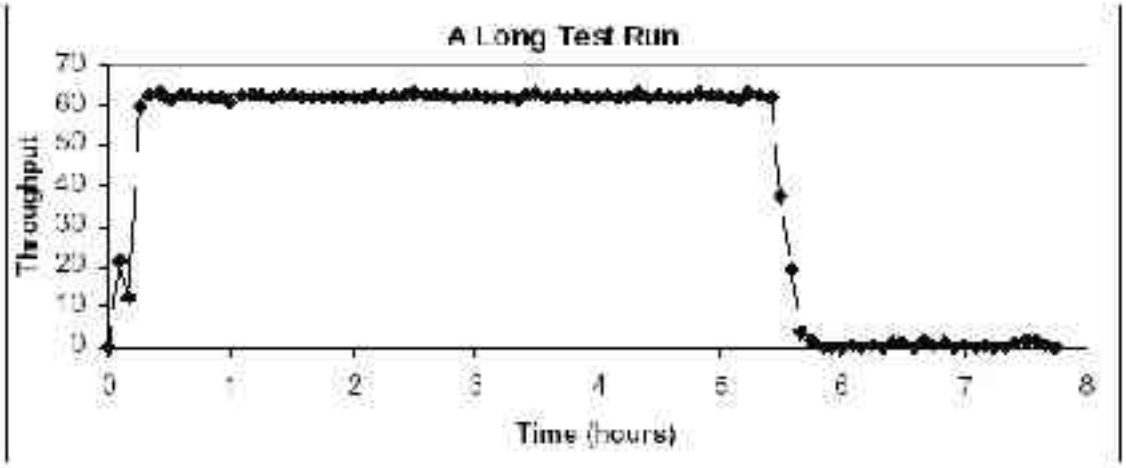


Figure 1.1 – Throughput vs. Time for Application with Fatal Memory Leak

Figure 1.2 – Memory Usage vs. Time for Application with Fatal Memory Leak

- Two charts form a “signature” for a fatal memory leak

3. Performance Modeling

- Estimation of end-to-end delay requires queueing network models
  - Only simple models need be used, because of the unpredictability of service components
- End-to-end delay/capacity analysis requires modeling of hardware and software resources
  - Layered queuing network approach is needed
- Desirable to have “standard” specification methods converted into queuing network models

Existing Approaches

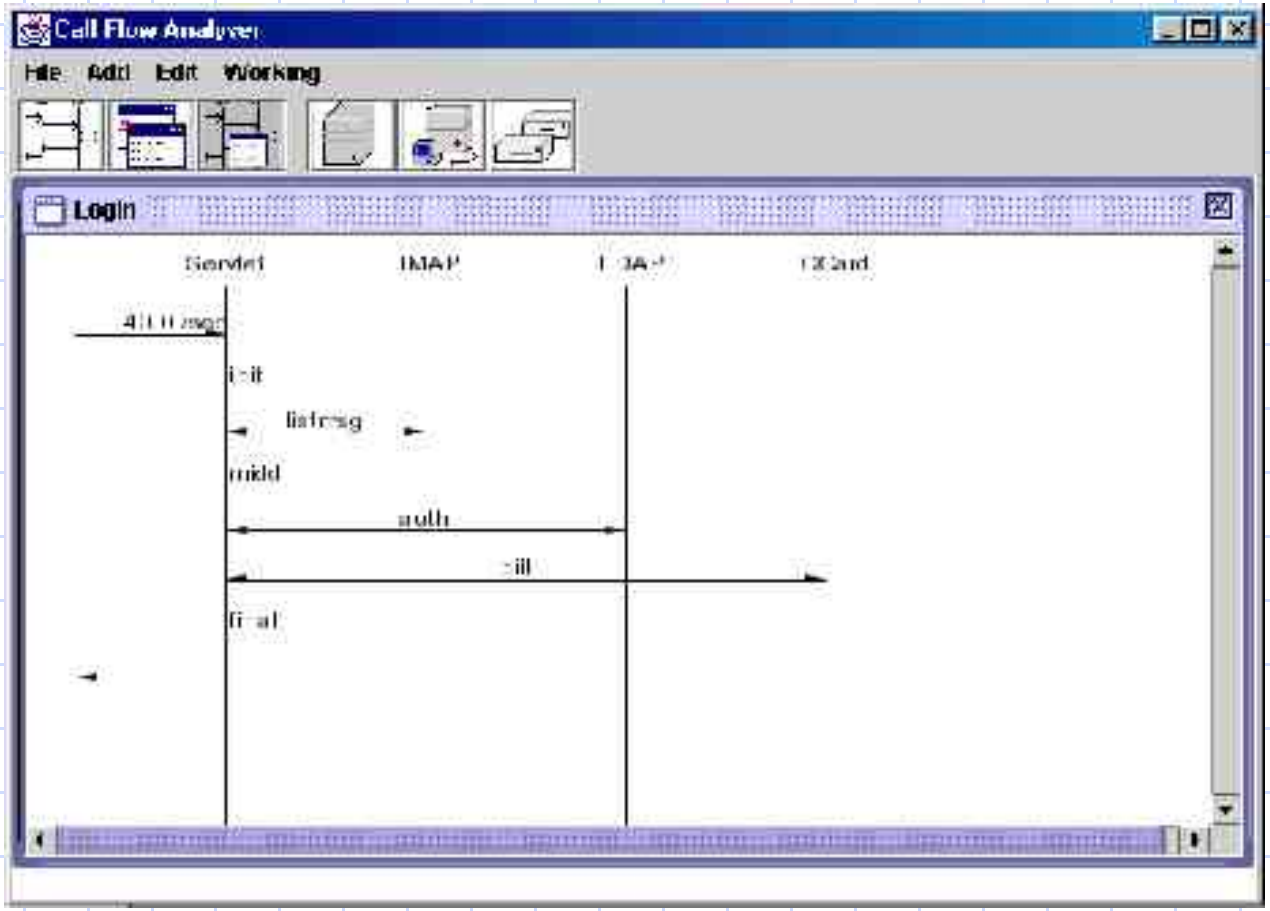
- Various tools and models for distributed system modeling – using a “layered approach”
  - Tool: Spe*ed[7]
    - Queueing network model generation from a software model specification, both hardware/software resources are specified
  - Layered Queueing Networks (M. Woodside et al)[5]
    - Generated from Use Case Maps, similar
  - Method of Layers (Roila, Sevcik)[4]

Ongoing Work at IIT-B

- A tool for performance analysts
- Should be simple
- Should have intuitive specification
- Should do simple models
  - Take away repetitive tasks from performance analyst
  - Leave advanced tasks to performance analyst

CFA- Call Flow Analyzer

Joint work with Mohit Gupta, now with Tata Consultancy Services.

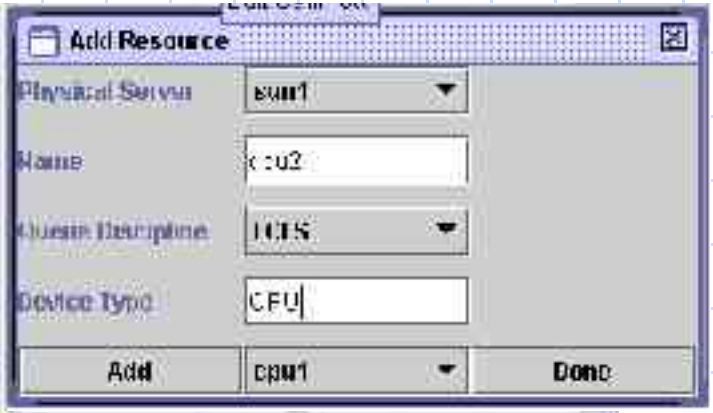


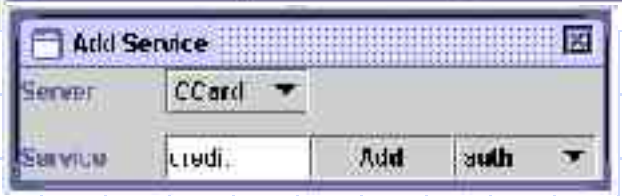
- Specification based on "call flow"
- Currently, simple calculations based on approximate open queueing models

CFA- Call Flow Analyzer

- Intuitive specification ↙ analytical solution
- Layered model
  - Software servers executing on hardware server
  - Hardware resources can be specified separately (server uses *x* ms on CPU, *y* ms on Disk)
  - Simple model of network links also included

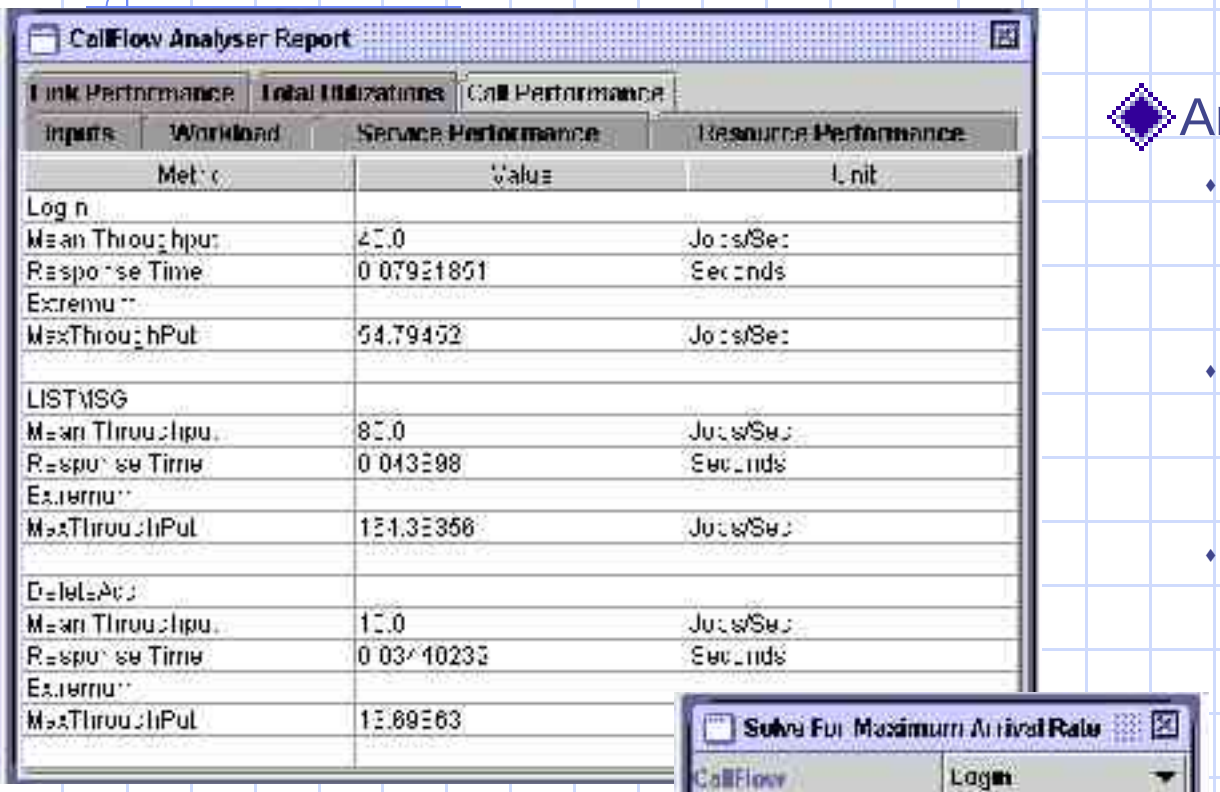
CFA- Call Flow Analyzer*





Developed by Mohit Gupta, TCS

CFA- Call Flow Analyzer



- Analysis results in
  - End-to-end response times of each user request
  - Maximum possible throughputs for each “call-flow”
  - Also computes maximum supportable arrival rate under average response time constraint

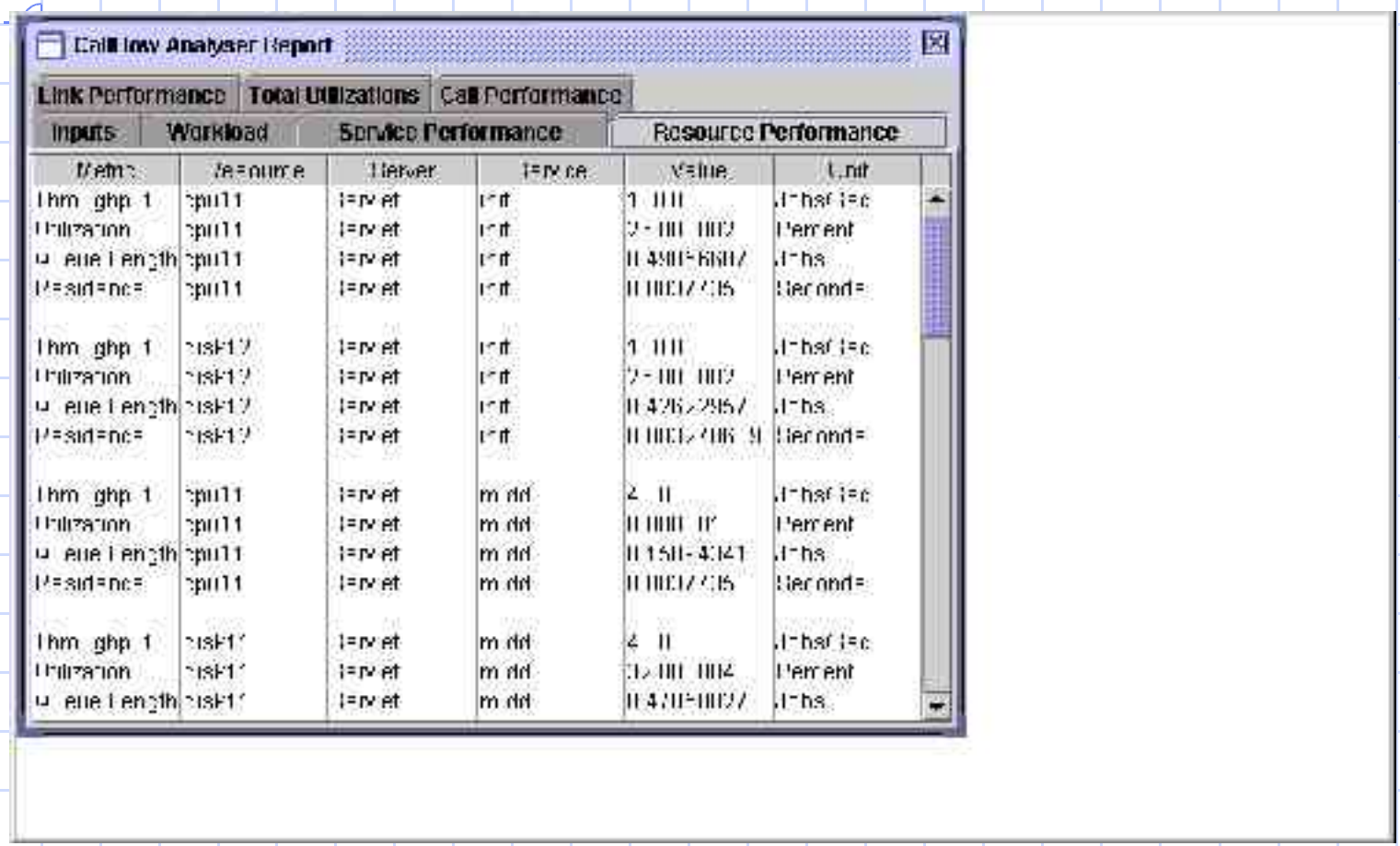
Summary

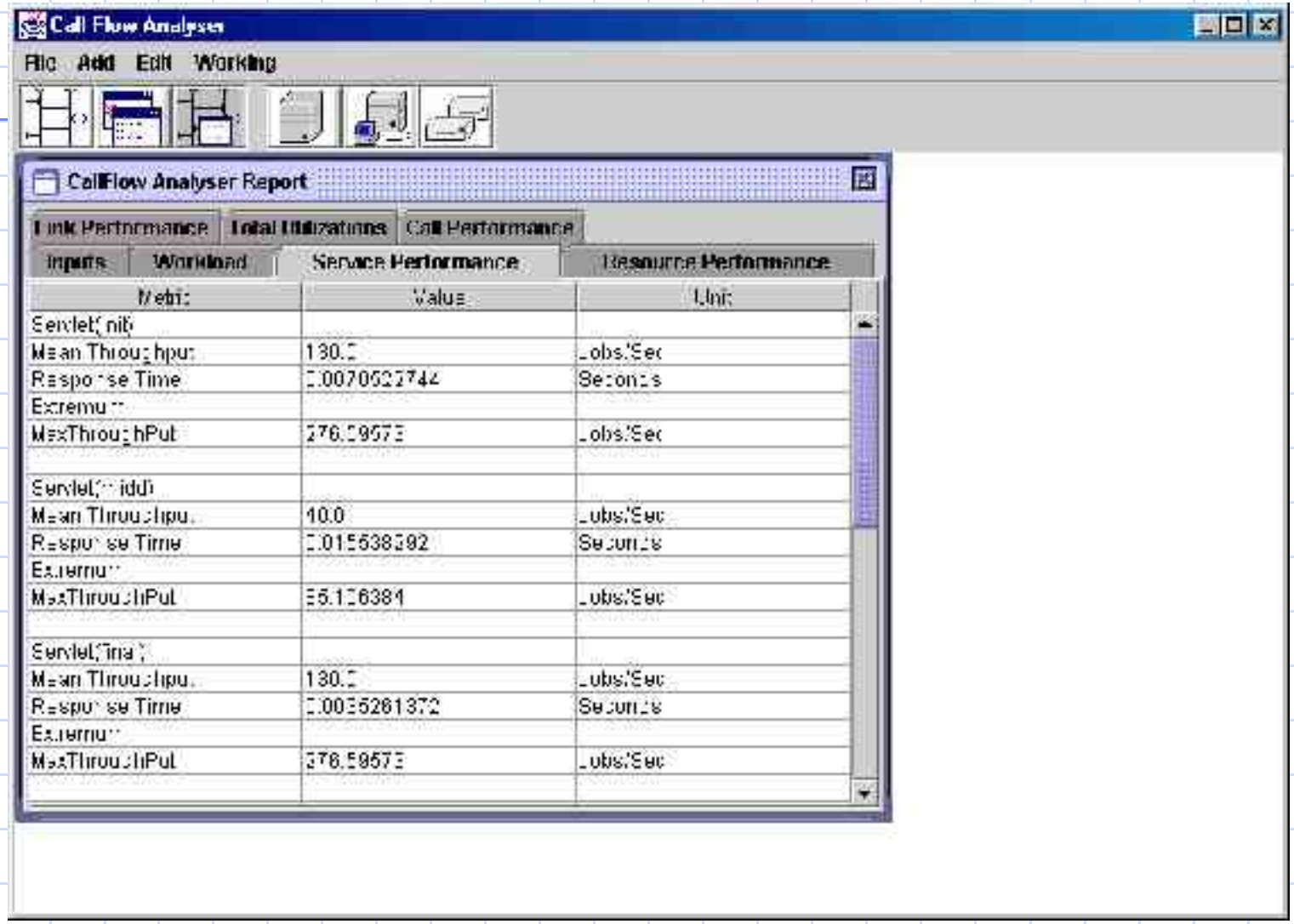
- Service performance engineering has significantly different challenges than those of product performance engineering
  - Many are not traditional queuing theory problems
  - Focus should be on available means and relevant analyses – this shifts focus to measurement tools, and tools that translate intuitive specifications to simple models
  - More work necessary on understanding how to analyze a gray box based on operating system measurements (some patent-pending work done in AT&T labs)

References

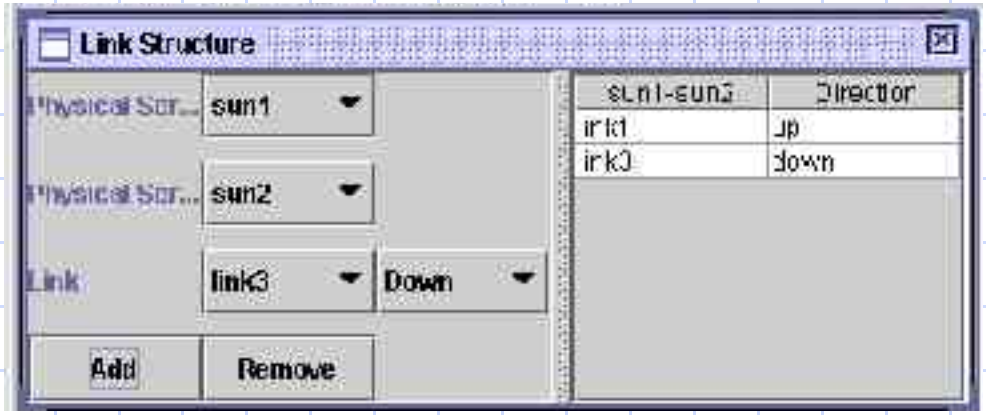
- P.K. Reeser, R. D. van der Meri, R. Hariharan, “An Analytic Model of a Web Server”, ITC-16, 1999.
- R. Hariharan, W.K. Ehrlich, D. Cura, P.K. Reeser, “End-to-end modeling of Web Server Architectures”, ACM Conference on Performance Analysis of Web Servers, 2000.
- A. Avritzer, R. Farel, K. Futamura, M. Hosseini-Nasab, A. Karasaridis, K. Meier-Hellstern, P. Reeser, P. Wirth, F. Hubner, D. Lucantoni, “ Internet Application Performance: A Signature-Based Empirical Approach”, in ITC-18, 2001.
- J.A. Rolia and K.C. Sevcik, "The Method of Layers", *IEEE-TSE*, SE-21, 8 (August 1995), 689-700.
- Dorin Petriu, Murray Woodside, "Software Performance Models from System Scenarios in Use Case Maps", Proceedings of Performance TOOLS 2002, London, April 2002
- V. Mainkar, “A Model of a Web Server with Dynamic Content”, INFORMS Fall 1999 Meeting, Philadelphia, PA.
- “SPE*ED – The Software Performance Engineering Tool”, http://www.perfeng.com/

Back-up slides


Call Inv Analyser Report
Link Performance
Total Utilizations
Call Performance
Inputs
Workload
Service Performance
Resource Performance
Metric
Resource
Server
Service
Value
Unit

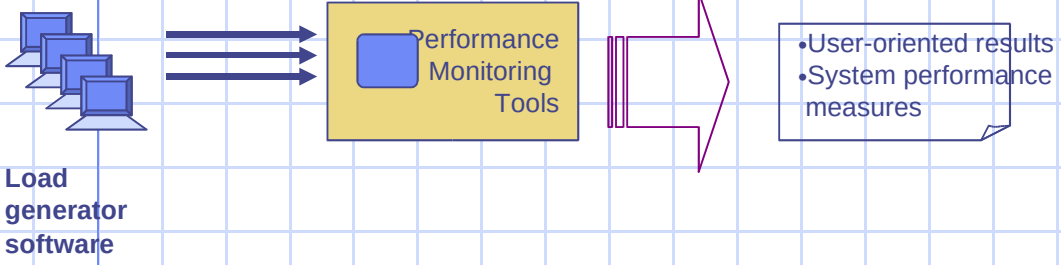

Call Flow Analyser
File Add Edit Working
CallFlow Analyser Report
Link Performance
Initial Utilizations
Call Performance
Inputs
Workload
Service Performance
Resource Performance
Value
Unit
Mean Throughput
Response Time
Extremum
MaxThroughPut
Mean Throughput
Response Time
Extremum
MaxThroughPut
Mean Throughput
Response Time
Extremum
MaxThroughPut
130.0
Jobs/Sec
Seconds
276.59573
Jobs/Sec
10.0
Jobs/Sec
Seconds
55.126381
Jobs/Sec
130.0
Jobs/Sec
Seconds
276.59573
Jobs/Sec

CFA- Call Flow Analyzer



Performance Measurement – *Tools*

Ongoing work at IIT-B

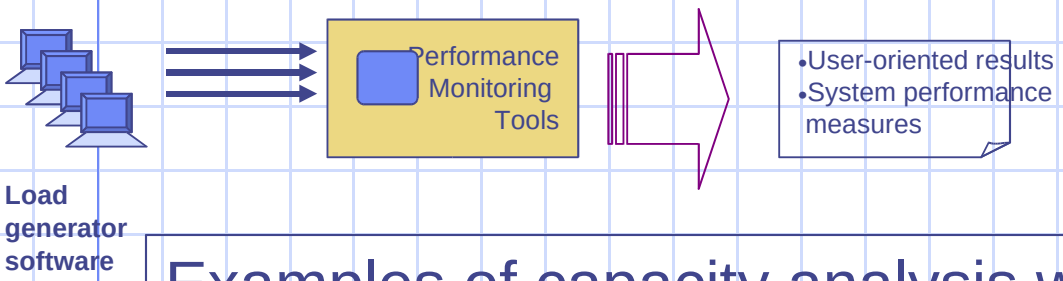


Examples of co-ordination work: Consider a load generator which is running in a mode in which it increases the load level every 10 minutes. The tool can do two types of tasks:

- ***Routine:*** **e.g. automatically mark data collected on the server side so that the corresponding load level can be identified**
- ***Intelligent:*** **e.g. figure out how long a duration of test is necessary to get "steady-state" results**

Performance Measurement – *Tools*

Ongoing work at IIT-B

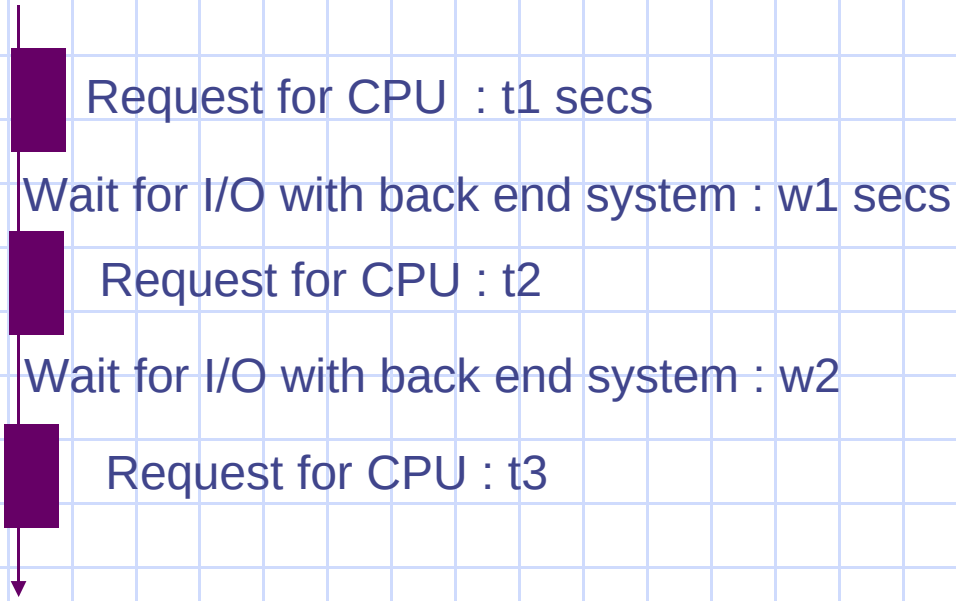


Examples of capacity analysis work:

- *Routine* calculations:
  - Load level (number of users, request rate, resource utilizations…) at which some performance requirement is met.
  - Generating graphs of throughput vs number of users, response time vs throughput, etc.
- *Intelligent* calculations: "knee" of response time curve, where does throughput curve flatten out…

Queueing Model : CPU

Flow of typical servlet that generates dynamic content :



CPU modeled as a processor sharing queue
Arrival rate of requests to this queue = Web transaction throughput rate X number of CPU request segments in the servlet

Hierarchical Queueing Model

Then, holding time of servlet is =

$w1 + w2 + \ldots.$

$+ R_{cpu}(t1) + R_{cpu}(t2) + R_{cpu}(t3) + \ldots$

where $R_{cpu}(t)$ is the response time of a request in the CPU queue

Model variables are interdependent, so iterate until convergence is achieved.