Performance Analysis of Computer Systems

Requirements, Metrics, Techniques, and Mistakes

Holger Brunst (holger.brunst@tu-dresden.de)
Matthias S. Mueller (matthias.mueller@tu-dresden.de)
Summary of Previous Lecture

Introduction and Motivation

Holger Brunst (holger.brunst@tu-dresden.de)
Matthias S. Mueller (matthias.mueller@tu-dresden.de)
Recommended Literature

- Raj Jain: *The Art of Computer Systems Performance Analysis*  
- Rainer Klar, Peter Dauphin, Fran Hartleb, Richard Hofmann, Bernd Mohr, Andreas Quick, Markus Siegle  
  Messung und Modellierung paralleler und verteilter Rechensysteme  
Announcements

Exercise leaf is on the web

Exercise tomorrow:
- 13:00 at INF E046 (?)
- Discussion of solution of previous exercise
- Questions regarding current exercise
Invent of computer changed our lives regarding data processing
Can do computations much faster. Factor: $10^{10} - 10^{14}$
Interplay of memory, cache, registers, and I/O
HPC: A key technology? Its role in the USA, Japan, Europe, and China
Required know how for HPC: Algorithms, performance, programming, and machine operation
Challenges: Languages, parallelization, network, scheduling, system architecture, programming models
Software is a key technology
ZIH is a Central Scientific Unit at Technische Universität Dresden

Support *real users*

Development of algorithms and methods in close cooperation with all departments

Provides HPC infrastructure and IT services

Research in: Performance analysis, programming methods, software tools, modeling of biological algorithms, mathematical models and algorithms

Role of mediator between vendors, developers, and users

Teaching and education
Previous Lecture: HPC Infrastructure

HPC-Komponente
Hauptspeicher 6,5 TB

PC-Farm

HPC-SAN
Festplattenkapazität: 68 TB

PC-SAN
Festplattenkapazität: 51 TB

PetaByte-Bandarchiv
Kapazität: 1 PB

8 GB/s

4 GB/s

1,8 GB/s

4 GB/s
Performance Analysis of Computer Systems

Requirements, Metrics, Techniques, and Mistakes

Holger Brunst (holger.brunst@tu-dresden.de)
Matthias S. Mueller (matthias.mueller@tu-dresden.de)
Outline

- Preliminary remarks
- Systematic approach to performance evaluation
- Metrics
- Comparison of evaluation techniques
- Common mistakes
The development of computer systems in respect of hard- and software is accompanied by performance measurements and modeling since the 1960s.

However, only a small fraction of the research work is applied in the field.

Ferrari (1986): The study of performance evaluation as an independent subject has sometimes caused researchers in the area to lose contact with reality.

Why is it that performance measurements are by no means an integrated and natural part of computer system development?

- The primary duty of system developers is to create functionally correct systems!
- Performance evaluation tends to be optional. Some people compare it to the freestyle event in ice-skating.
The term *performance* is ambiguous in computer science. It can stand for:

- “Well, it’s functioning (more or less)”;
- A short development cycle;
- High throughput;
- Short response times;
- Good reliability.

Doherty (1970)
Performance is the degree to which a computing system meets expectations of the persons involved in it.

Graham (1973)
Performance ... is the effectiveness with which the resources of the host computer system are utilized toward meeting the objectives of the software system. Or short:
How well does the system enable me to do what I want to do?
Performance Evaluation

- Ferrari (1978)
  We use the term 'performance' to indicate how well a system, assumed to perform correctly, works.

- DIN-Norm 66273
  The German DIN-Norm considers a computer system as a black box and „... baut die Messung und Bewertung der Schnelligkeit ausschließlich auf das Verhalten der Datenverarbeitungsanlage an der vom Anwender sichtbaren Schnittstelle auf.“

- Jain (1991)
  Contrary to common belief, performance evaluation is an art. ... Like artist, each analyst has a unique style. Given the sample problem, two analysts may choose different performance metrics and evaluation methodologies.
Objectives

- Performance analysis: Get highest performance for a given cost
- "Performance Analyst": Anyone who is associated with computer systems, i.e. systems engineers and scientists but also users
- Which tasks need to be carried out?
- Tasks:
  - Specification of performance requirements
  - Evaluation of design alternatives
  - Comparison of two or multiple systems
  - Finding the best value of given system parameter (system tuning)
  - Identification of bottlenecks
  - Workload characterization for a given system
  - Finding the right size and number of hardware and software components (capacity planning)
  - Performance prediction at future workloads (forecasting)
Conventions

- **System:**
  - An arbitrary collection of hardware, software, and firmware e.g.:
    - CPU
    - Database
    - Network of computers

- **Metric:**
  - A criteria used to evaluate the performance of a system e.g.:
    - Response time
    - Throughput
    - Floating point operations per second

- **Workload(s):**
  - Representative collection of user requests to a system e.g.:
    - CPU workload: Instructions to execute
    - Database workload: Which queries to perform
Example 1: Select Metric, Technique, Workload

What performance metrics should be used to compare the performance of disk drives or SANs?

How and where would you start?

Examples:

- Capacity
- Price
- Read/write throughput
- Seek latency
- Energy consumption
- Mean-time to failure
- Emission of heat and noise
- Form factor etc.
Example 2: Correctness of Perf. Measurements

How to measure the performance of a computer system?
At least two tools are required:
- Load generator
- Performance monitor

Which type of monitor would be more suitable (software or hardware) for measuring the following quantities?
- Number of instructions executed by a processor
- Degree of multiprogramming on a timesharing system
- Response time of packets on a network
Example 3: Experiment Design

The performance of a system depends on the following three factors:

- Garbage collection used: G1, G2, or none
- Type of workload
  - Editing
  - Computing
  - Artificial intelligence
- Type of CPU: C1, C2, or C3

How many experiments are needed?
How does one estimate the performance impact of each factor?
Example 4: Simple Queuing Models

The average response time of a database system is three seconds. During a 1-minute observation interval, the idle time on the system was 10 seconds.

A queuing model for the system can be used to determine the following:

- System utilization
- Average service time per query
- Number of queries completed during the observation interval
- Average number of jobs in the system
- Probability of number of jobs in the system > 10
- 90 percentile response time $t$
  - 90% of observations stay below $t$
  - German: Perzentile/Prozentränge oder allg.: Quantile
  - Reminder: 50$^{th}$ percentile is the median
- 90 percentile waiting time
The Art of Performance Evaluation

- Successful evaluation cannot be produced mechanically
- Evaluation requires detailed knowledge of the system to be modeled
- Careful selection of methodology, workload, and tools
- Conversion from an abstract feeling or notion to a real problem which needs to be formalized in a way that can be handled by established tools
- Analysts tend to have different "styles"
Systematic Performance Evaluation (1)

TEN STEPS:

1. **State goals** of the study and define the system
   - Identical hardware and software: Yet, the system may vary depending on goals
   - The chosen system boundaries affect the performance metrics as well as the workloads used to compare the systems
   - Additionally: Administrative control of the sponsors of the study. Sponsors may want to keep uncontrollable components out of the system boundaries

2. **List services** and outcomes
   - Network: Send packets to a specified destination
   - Processor: Perform a number of different instructions
   - Database: Respond to queries
   - Also list the possible outcomes, e.g. db query: correctly, incorrectly, not at all

3. **Select metrics**
   - Criteria to compare the performance: usually **speed**, **accuracy**, and **availability**
     - Network: throughput, delay (speed); error rate (accuracy)
     - CPU: time to execute various instructions (speed)
4. **List parameters** that affect performance
   - System parameters (both hardware and software)
   - Workload parameters (characteristics of users’ requests)
   - The list of parameters may not be complete
   - Parameters may be added, always keep list as comprehensive as possible

5. **Select factors** to study
   - Factors: Parameters that are varied during the evaluation
   - **Levels**: Values of a factor
   - Limited resources ➔ start with a short list and extend if the resources permit
   - Chose parameters expected to have high impact as factors
   - Also consider economic, political, technological constraints, and decision makers

6. **Select technique** for evaluation
   - **Analytical modeling, simulation, measuring** a real system
   - Depends on time, resources, and the desired level of detail
7. Select workload
   - List of service requests to the system
   - Depends on the evaluation technique: probability of various requests (analytical), trace of requests from real system (simulation), user scripts (measurement)
   - Representative workloads often require to measure and characterize the workload on existing systems

8. Design experiments
   - Maximum information with minimum effort
   - Two phases:
     - First: Many factors, only few levels \(\rightarrow\) determine relative effect of factors
     - Second: Few most significant factors, increase the number of levels

9. Analyze and interpret data
   - Consider the variability of simulation and measurement results. Use statistics!
   - Interpretation is the key part of the analyst: Analysis produces results but no conclusions or decisions
   - Analysts’ conclusions may be different given the same set of results
10. Present results:

- Communicate the results to other member of the decision-making team
- Information needs to be easily understood
  - No statistical jargon!
  - Chose graphic form with proper scaling of graphs
- At this point: Reconsider and question some of the decisions made in the previous steps (e.g. system boundaries, factors, or metrics)
- The complete evaluation project consists of several cycles rather than a single sequential pass
Performance Metrics

What is a performance metric?
- The absolute number a service has been carried out
- The time taken to perform a service
- The size of the resources required to perform a service

Two options
- Use values directly
- Normalize values to a common time basis to provide a speed metric (divide number by time)
- Derive probabilities

Choosing an appropriate performance metric depends on the goals and the costs of the performance study
Performance Metrics by Service Outcome

System

Service Request $i$

Correctly?

yes

Time (Response time)
Rate (Throughput)
Resource (Utilization)
Probability
Time between errors
Duration of the event
Time between events

no

Can do?

yes

Error $j$

no

Event $k$

LARS: Requirements, Metrics, Techniques
Characteristics of Good Performance Metrics

- Easiness of measurements
  - If a metric is not easy to measure, it is unlikely that anyone will actually use it
  - If it is complicated, it might be much more difficult to measure the metric correctly

- Consistency
  - Units of the metric and its precise definition are the same across different configurations and different systems
  - Not true in many cases (ex. MIPS and MFLOPS)

- Independence
  - Commonly used metrics are often used for decisions to select a system
  - Good metric should be independent from intervention from vendors to influence the composition of the metric to their benefit
Commonly Used Performance Metrics (1)

- Clock rate
  - Most prominent indication of performance often is the frequency of the processors central clock
  - This performance metric completely ignores how much computation is actually performed
  - It is repeatable, easy to measure, consistent, no games from vendors, but ...
  - It is nonlinear and unreliable

- Number of cores!

- MIPS
  - Millions Instructions per Second
  - Rate metric (amount of computation performed per time unit)
  - It is easy to measure, repeatable, independent, but
  - Nonlinear, not reliable, and not consistent
  - problem: amount of computations per instruction differ (also: RISC, CISC)
Commonly Used Performance Metrics (2)

- **FLOPS**
  - Floating Point Operations per second (Mega-, Giga-, TeraFLOPS)
  - Defines an arithmetic operation on two floating point quantities to be the basic unit
  - Tries to correct shortcoming of the MIPS metric
  - No value for integer applications
  - Agreeing on exactly how to count the number still difficult
  - Pretty much the dominant metric in the HPC field
  - It is repeatable, easy to measure (now), but ...
  - It is nonlinear and inconsistent, there are some games from vendors

- **SPEC**
  - Standard Performance Evaluation Cooperative (SPEC)
  - Collection of specialized benchmarks (e.g. CINT2006, CFP2006, etc.)
Commonly Used Performance Metrics (3)

- QUIPS (QUality Improvement Per Second)
  - Traditionally: Metrics define effort to reach a certain result
  - Here: Metric defines the quality of a solution
  - Quality is defined based on mathematical characteristics of a given problem
  - Personal impression: not relevant (at least not in HPC)

- Execution time (system/user)
- Wall clock time
Commonly Used Performance Metrics (4)

- **Response time**
  - The time interval between a user’s request and the system response
  - Response time, reaction time, turnaround time, etc.
  - Small response time is good:
    - For the user: waiting less
    - For the system: free to do other things

- **Throughput**
  - Number of work units done per time unit
  - Applications being run, files transferred, etc.
  - High throughput is good
    - For the system: was able to serve many clients
    - For the user: might imply worse service
  - MIPS is one measure of throughput
Commonly Used Performance Metrics (5)

Utilization
- Percentage of time the system is busy serving clients
  - Important for expensive shared system
  - Less important (if at all)
  - for single user systems, for real time systems
- Utilization and response time are interrelated
  - Very high utilization may negatively affect response time

Other metrics:
- Mean Time Between Failures (MTBF)
- Supportable load
- Speedup
- Scalability (weak/strong)
Evaluation Techniques: Analytical Modeling

- Based on a rigorous mathematical model
- Provides the best insight into the effects of different parameters and their interaction
  - Is it better to configure the system with one fast disk or with two slow disks?
- Can be done before the system is built and takes a short time
- Rarely accurate
  - Usually needs many simplifying assumptions
  - Depends on the quality and correctness of these assumptions
Evaluation Techniques: Simulation

- Simulate the system operation (usually only small parts thereof)
- Flexibility: full control of simulation model, parameters, level of detail
- Disk: average seek time vs. acceleration and stabilization of the head
- Can be done before the system is built
  - Simulation of a full system is infeasible
  - Simulation of the system parts does not take everything into account
Evaluation Techniques: Measurement

- Implement the system in full and measure its performance directly
- The most convincing
  - Effects of varying parameter values cannot (if at all) be easily isolated
  - Often confused with random changes in the environment
- High cost:
  - Implement the system in full, buy hardware
## Evaluation Techniques: Pros and Cons

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Analytical Modeling</th>
<th>Simulation</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage</td>
<td>Any</td>
<td>Any</td>
<td>Post-prototype</td>
</tr>
<tr>
<td>Time Required</td>
<td>Small</td>
<td>Medium</td>
<td>Varies</td>
</tr>
<tr>
<td>Tools</td>
<td>Analysts</td>
<td>Computer languages</td>
<td>Instrumentation</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Low</td>
<td>Moderate</td>
<td>Varies</td>
</tr>
<tr>
<td>Trade-off evaluation</td>
<td>Easy</td>
<td>Moderate</td>
<td>Difficult</td>
</tr>
<tr>
<td>Cost</td>
<td>Small</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Saleability</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>
The Bottom Line

- Simulation is the most widely used technique
- Combination of techniques is recommended
- Never trust the results produced by the single method
- Validate with another one, e.g.
  - analysis + simulation
  - simulation + measurements,
Common Mistakes in Performance Analysis

- None
- Biased
- Problem?

- Goals
  - None
  - Biased
  - Problem?

- Methodology
  - Unsystematic
  - Incorrect Metrics
  - Bad Workload
  - Wrong Technique
  - Bad Experiments

- Completeness
  - Overlooking Parameters
  - Ignore Factors
  - Wrong Level of Detail

- Analysis
  - None
  - Erroneous
  - Too Complex
  - No Sensitivity
  - Ignoring Input Errors
  - Ignoring Variability
  - No Outlier Handling

- Presentation
  - Improper Presentation
  - Ignoring Social Aspects
  - Omitting Assumptions
  - Omitting Limitations

LARS: Requirements, Metrics, Techniques
Common Mistakes: What are the goals?

- No goals with a good understanding of the problem
  - Many performance efforts are started without clear goals
  - Performance model must be developed with a particular goal in mind
  - First, understand the system and the problem (40%)
  - Then, start writing the simulation code
  - Not trivial. Goals often change with a better understanding of the problem

- Biased goals
  - “show that one system is better than another”
  - Metric and workload are not selected for proper comparison but for highlighting a given system
  - Performance analysts are to be unbiased!
  - The role of a performance analyst is like that of a jury
  - Depend your conclusions on results rather than on believes
Common Mistakes: Methodology Selection

- Unsystematic approach
  - Arbitrary selection of system parameters, factors, metrics, and workloads lead to inaccurate conclusions. Be complete!

- Incorrect performance metrics
  - Example 1: Comparison of MIPS of a RISC and a CISC architecture
  - Example 2: Computer advertisement “datasheets” for GHz, GB, Core number, and Megapixel fans

- Unrepresentative workload
  - Workload should represent the actual usage of the system in practice
  - Example: Packet sizes in a network

- Wrong evaluation technique
  - Analysts are often “married” with one technique, i.e. measurement, or simulation, or analytical modeling
  - Resulting in model optimized for the analyst rather than the problem
  - An analyst should have a basic knowledge of all three techniques
Common Mistakes: Completeness and Balance

- Overlooking important parameters
  - List system and workload characteristics that affect performance
  - System: quantum (CPU) and working set (memory) size
  - Workload: number of users, request patterns, priorities

- Inappropriate level of detail
  - Very different alternatives: Use high-level model
  - Slight variations: Use more detailed model
  - Do not take a detailed approach when a high-level model will do and vice versa

- Ignoring significant factors
  - Varied parameters are called factors
  - Usually, not all parameters are factors.
  - Identify the ones that significantly alter performance if varied e.g. response time: packet size vs. arrival rate
  - Favor factors that are directly controlled by the user
  - The choice of factors should be based on relevance, not on their knowledge
Common Mistakes: Analysis

- No analysis
  - Analysts are good at collecting enormous amounts of data but often cannot analyze the data and write understandable summaries
  - Result: No useful analysis at all or a thick report with many graphs but no interpretation
  - Teamwork can help

- Erroneous analysis
  - Let’s average ratios! Short simulation runs or so much more convenient!

- No sensitivity analysis (German: Empfindlichkeitsanalyse)
  - Do not present your results as facts but as evidence
  - Performance results may be sensitive to workload and system parameters

- Ignoring errors in input
  - Parameters of interest cannot be measured. Example: Network device

- Improper treatment of outliers: Measurement error vs. system phenomenon

- Analysis too complex: Published models are often too complex for the real world

- Ignoring variability: Common to analyze only the mean performance. Example: Daily averages of computer demands which ignore the large hourly peaks.
Common Mistakes: Presentation

- Improper presentation of results
  - Help decision making
  - “The right metric to measure the performance of an analyst is not the number of analyses performed but the number of analyses that helped the decision makers.”

- Ignoring social aspects
  - Presentation requires social and substantive skills!
  - Analysts typically have good substantive skills...
  - Trust between analyst and decision makers
  - Conflict of interest: Innovativeness of the modeling approach (analyst) vs. quickly getting to the final results (decision maker)

- Omitting assumptions and limitations
  - Users will try to reproduce your results under their assumptions which is likely to reveal different results
Checklist for Avoiding Mistakes I

- Is the **system correctly defined** and the **goals clearly stated**?
- Are the goals stated in an **unbiased** manner?
- Have all the steps of the analysis followed **systematically**?
- Is the **problem clearly understood** before analyzing it?
- Are the performance **metrics relevant** for this problem?
- Is the **workload correct** for this problem?
- Is the **evaluation technique** **appropriate**?
- Is the list of **parameters** that affect performance **complete**?
- Have all **parameters** that affect performance been chosen as factors to be **varied**?
- Is the **experimental design efficient** in terms of time and results?
- Is the **level of detail** proper?
- Is the measured data presented with **analysis and interpretation**?
Checklist for Avoiding Mistakes II

- Is the analysis **statistically correct**?
- Has the **sensitivity analysis** been done?
- Would **errors in the input** cause an insignificant **change in the results**?
- Have the **outliers** in the input or output been **treated properly**?
- Have the **future changes** in the system and workload been modeled?
- Has the **variance of input** been taken into account?
- Has the **variance of the results** been analyzed?
- Is the **analysis easy** to explain?
- Is the **presentation style suitable** for its audience?
- Have the results been **presented graphically** as much as possible?
- Are the **assumptions and limitations** of the analysis clearly documented?
Short Example: Bandwidth to Filesystems

- State goals and define the system
  - read and write with 8 GB/s.
  - move 25 TB in less than 4h
- List services and outcomes
  - File system
- Select metrics
  - Bandwidth in GB/s
- List parameters
  - Block size, Number of clients, Total data written, type of I/O (buffered, direct)
- Select factors to study
- Select evaluation technique
  - Measurement
- Select workload
  - 7/8 of memory, 25 TB of data
- Design experiments
- Analyze and interpret data
- Present results
Structure of I/O Benchmark

start jobs

io_bandwidth #0 → io_bandwidth #1 → io_bandwidth #2 → ... → io_bandwidth #n

n chunks written

write 1

create file i_1

write chunk

time stamp

write chunk

time stamp

write chunk

time stamp

write 1

write 2

call jobs

read 1

read 2

MPI_Gather all timestamps

summary

to stdout

#0 → #1 → #2 → ... → #n

#0 → #1 → #2 → ... → #n

#0 → #1 → #2 → ... → #n

LARS: Requirements, Metrics, Techniques
Local Caching and Fairness

fair benchmark: same file size for all nodes

1 GB for OS
allocate and touch the rest of the memory
allocated, but not written

write memory to file
const. file size

1 GB for OS
allocate and touch the rest of the memory
allocated, but not written

write memory to file
const. file size
Thank You!

Holger Brunst (holger.brunst@tu-dresden.de)
Matthias S. Mueller (matthias.mueller@tu-dresden.de)