Performance Evaluation of Computer Systems

Jean-Marc Vincent and Arnaud Legrand

POLARIS Project
Laboratory of Informatics of Grenoble (LIG)
Université Grenoble-Alpes
{Jean-Marc.Vincent,Arnaud.Legrand}@imag.fr
https://team.inria.fr/polaris/
http://mescal.imag.fr/membres/arnaud.legrand/teaching/2016/

2016
Outline

1. Scientific context
2. Methodology
3. Performance indexes
Outline

1 Scientific context

2 Methodology

3 Performance indexes
Research activities in performance evaluation

Teams in Grenoble

- Polaris project: Large systems (clusters and grids)
- DataMove project: Interactive parallel systems
- Drakkar team: Networking
- Erods (Sardes): Middleware
- Verimag: Embedded systems
- etc

Industrial collaborations

- Orange-Lab: load injectors, performances of middlewares
- HP-Labs: cluster computing, benchmarking
- Bull: benchmarking, performances analysis
- ST-Microelectronics
Research activities in performance evaluation

Teams in Grenoble
- Polaris project: Large systems (clusters and grids)
- DataMove project: Interactive parallel systems
- Drakkar team: Networking
- Erods (Sardes): Middleware
- Verimag: Embedded systems
- etc

Industrial collaborations
- Orange-Lab: load injectors, performances of middlewares
- HP-Labs: cluster computing, benchmarking
- Bull: benchmarking, performances analysis
- ST-Microelectronics
Complexity of computer systems

- **hierarchy**: level decomposition: OS / Middleware / Application
- **distribution**: asynchronous resources: memory, CPU, network
- **dynamicity**: architecture and environment (reliability, mobility, ...)
- **scalability**: number of components (autonomous management)

Typical problems

- Minimize losses in routing policies
- Minimize active waiting in threads scheduling
- Maximize cache hits
- Optimise block sizes in parallel applications
- Maximize throughput of communication systems
- Fix time-outs, reemission periods, ...
- Fix the granularity: pages, blocks, tables, message sizes...
- ...

Application context (1)

Complexity of computer systems

- **hierarchy**: level decomposition: OS / Middleware / Application
- **distribution**: asynchronous resources: memory, CPU, network
- **dynamicity**: architecture and environment (reliability, mobility,...)
- **scalability**: number of components (autonomous management)

Typical problems

- Minimize losses in routing policies
- Minimize active waiting in threads scheduling
- Maximize cache hits
- Optimise block sizes in parallel applications
- Maximize throughput of communication systems
- Fix time-outs, reemission periods, ...
- Fix the granularity: pages, blocks, tables, message sizes...
- ...

Scientific context

Methodology

Performance indexes
Application context (2)

Typical “hot” applications
- **Peer to peer systems**: dimensionning, control
- **Mobile networks**: ad-hoc networking, reactivity, coherence
- **Grids**: resources utilization, scheduling
- etc

Other application domains
- production systems: production lines, logistic,...
- embedded systems
- modelling of complex systems: biology, sociology,...
- etc
Typical “hot” applications

- **Peer to peer systems**: dimensionning, control
- **Mobile networks**: ad-hoc networking, reactivity, coherence
- **Grids**: resources utilization, scheduling
- etc

Other application domains

- production systems: production lines, logistic,...
- embedded systems
- modelling of complex systems: biology, sociology,...
- etc
Outline

1. Scientific context
2. Methodology
3. Performance indexes
Development of parallel/distributed applications

**Scientific context**

**Methodology**

**Performance indexes**

**Qualitative specifications**: Is the result correct?
- property verification: formal/automatic proofs
- testing: critical dataset

**Quantitative specifications**: Is the result obtained in a reasonable time?
- performance model
- performance measurements

**Problem identification**: localization of the problem
- debugging, log analysis
- performance statistical analysis

**Modification and control**
- source code / libraries / OS / architecture
- parameters of the system: dimensioning
- control algorithms: tuning
Development of parallel/distributed applications

Qualitative specifications: Is the result correct?
- property verification: formal/automatic proofs
- testing: critical dataset

Quantitative specifications: Is the result obtained in a reasonable time?
- performance model
- performance measurements

Problem identification: localization of the problem
- debugging, log analysis
- performance statistical analysis

Modification and control
- source code / libraries / OS / architecture
- parameters of the system: dimensioning
- control algorithms: tuning
Development of parallel/distributed applications

**Qualitative specifications**: Is the result correct?
- property verification: formal/automatic proofs
- testing: critical dataset

**Quantitative specifications**: Is the result obtained in a reasonable time?
- performance model
- performance measurements

**Problem identification**: localization of the problem
- debugging, log analysis
- performance statistical analysis

**Modification and control**
- source code / libraries / OS / architecture
- parameters of the system: dimensioning
- control algorithms: tuning
Development of parallel/distributed applications

**Qualitative specifications**: Is the result correct?
- property verification: formal/automatic proofs
- testing: critical dataset

**Quantitative specifications**: Is the result obtained in a reasonable time?
- performance model
- performance measurements

**Problem identification**: localization of the problem
- debugging, log analysis
- performance statistical analysis

**Modification and control**
- source code / libraries / OS / architecture
- parameters of the system: dimensioning
- control algorithms: tuning
Dual analysis

Understand the behavior of a distributed application

1. Identification of distributed patterns, states of the system
2. Pattern verification
3. Time evaluation
4. Global analysis of the execution and performance synthesis
5. System monitoring
6. Global cost evaluation for the application user

Understand resources utilization

1. Hierarchical model of resources
2. Evaluation of utilization at:
   - Application level; executive runtime;
   - Operating system; hardware architecture
3. Global cost evaluation for the resources manager
Dual analysis

Understand the behavior of a distributed application

1. identification of distributed patterns, states of the system
2. pattern verification
3. time evaluation
4. global analysis of the execution and performance synthesis
5. system monitoring
6. global cost evaluation for the application user

Understand resources utilization

1. hierarchical model of resources
2. evaluation of utilization at:
   - application level; executive runtime;
   - operating system; hardware architecture
3. global cost evaluation for the resources manager
Methodology (1)

Complex system
Scientific context

Methodology

Performance indexes

Methodology (1)

Complex system

MODELISATION

Model

Formal Description Language
Methodology (1)

Complex system

MODELISATION

Model

Formal Description Language

METHODS

Computation

Model
Methodology (1)

Complex system

Model

Formal Description Language

Methods

Complexity

Computation

Analysis tools

Modelisation
Methodology (1)

- Complex system
- Model
  - Formal Description Language
- Measures
- VALIDATION
- EXPERIMENTATION
- METHODS
- COMPLEXITY
- Computation
  - Analysis tools

- MODELISATION
Methodology (1)

- **Complex system**
  - **Model**
    - Formal Description Language
  - **Measures**
    - Measurement tools
  - **Computation**
    - Analysis tools
  - **Validation**
  - **Methods**
  - **Parameters**
  - **Complexity**
  - **Experimentation**
  - **Modelisation**

Scientific context

Methodology

Performance indexes
Methodology (1)

Scientific context

Methodology

Performance indexes

MODELISATION

Complex system

MEASUREMENT

MEASURES

MEASUREMENT TOOLS

EXPERIMENTATION

COMPLEXITY

PARAMETERS

COMPUTATION

ANALYSIS TOOLS

CONTROL

MODEL

FORMAL DESCRIPTION LANGUAGE

METHODS

VALIDATION
## Evaluation methods

### From abstraction to physical reality

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
</table>

**Remarks:**
- Hybrid methods (emulation, load injectors, synthetic programs, ...)
- Dynamical process of evaluation
- Experimentation ⇒ Planning experiments methodology
Evaluation methods

From abstraction to physical reality

Model
Mathematical

Method
Analysis (formal, numerical, approximation)

Remarks:
- Hybrid methods (emulation, load injectors, synthetic programs,...)
- Dynamical process of evaluation
- Experimentation ⇒ Planning experiments methodology
Evaluation methods

From abstraction to physical reality

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>Analysis (formal, numerical, approximation)</td>
</tr>
<tr>
<td>Software</td>
<td>Simulation (discrete events)</td>
</tr>
</tbody>
</table>

Remarks:
Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology
## Evaluation methods

### From abstraction to physical reality

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>Analysis (formal, numerical, approximation)</td>
</tr>
<tr>
<td>Software</td>
<td>Simulation (discrete events)</td>
</tr>
<tr>
<td>Prototype</td>
<td>Observation (measures)</td>
</tr>
</tbody>
</table>

Remarks:

- Hybrid methods (emulation, load injectors, synthetic programs,...)
- Dynamical process of evaluation
- Experimentation $\Rightarrow$ Planning experiments methodology
Evaluation methods

From abstraction to physical reality

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>Analysis (formal, numerical, approximation)</td>
</tr>
<tr>
<td>Software</td>
<td>Simulation (discrete events)</td>
</tr>
<tr>
<td>Prototype</td>
<td>Observation (measures)</td>
</tr>
</tbody>
</table>

Remarks:

- Hybrid methods (emulation, load injectors, synthetic programs,...)
- Dynamical process of evaluation
- Experimentation ⇒ Planning experiments methodology
### Evaluation methods

#### From abstraction to physical reality

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>Analysis (formal, numerical, approximation)</td>
</tr>
<tr>
<td>Software</td>
<td>Simulation (discrete events)</td>
</tr>
<tr>
<td>Prototype</td>
<td>Observation (measures)</td>
</tr>
</tbody>
</table>

#### Remarks:

Hybrid methods (emulation, load injectors, synthetic programs,...)

Dynamical process of evaluation

**Experimentation ⇒ Planning experiments methodology**
Steps for a Performance Evaluation Study (Jain)

1. State the goals of the study: level of decision, investment, optimization, technical, ...
2. Define system boundaries.
3. List system services and possible outcomes.
4. Select performance metrics.
5. List system and workload parameters
6. Select factors and their values.
7. Select evaluation techniques.
8. Select the workload.
9. Design the experiments.
10. Analyze and interpret the data.
11. Present the results. Start over, if necessary.
Objective

1. Be able to analyze and predict performances of computer systems
2. Be able to build a software environment that produces the performances indexes.

Methods

1. Specification and identification of problems: modeling
2. Analysis of quantitative models: formal, numerical, simulation
3. Experimentation and statistical data analysis.


Aim of the course

**Objective**

1. Be able to analyze and predict performances of computer systems
2. Be able to build a software environment that produces the performances indexes.

**Methods**

1. Specification and identification of problems : modeling
2. Analysis of quantitative models : formal, numerical, simulation
3. Experimentation and statistical data analysis.
Organization of the course

Performance evaluation of systems 10 cours + 10 TD/TP
http://mescal.imag.fr/membres/arnaud.legrand/teaching/2013/RICM4_EP.php

Evaluation
Experimental project
Organisation

Team

Arnaud Legrand
Simulation for large scale systems analysis and control

Jean-Marc Vincent
Markovian modeling of systems, simulation and dimensioning
References : text books

  Covers the content of the course, a complete book

  Covers the statistical part of the course

- **Measuring Computer Performance: A Practitioner’s Guide** David J. Lilja *Cambridge University press 2000*
  Covers the practical part of measurement and benchmarking

  Covers the part on simulation
References: journals and conferences

- **General**: JACM, ACM Comp. Surv., JOR, IEEE TSE,...
- **Specialized**: Performance Evaluation, Operation research, MOR, ACM TOMACS, Queueing Systems, DEDS, ...
- **Application**: IEEE TPDS, TC, TN, TAC, Networks,...
- **Conferences on performances**: Performance, ACM-SIGMETRICS, TOOLS, MASCOT, INFORMS, ...
- **Conferences on an application domain**: ITC, Europar, IPDPS, Renpar, ...
- **National seminars**: Atelier d’évaluation de performances,...
Networking

Flow performance
- latency, waiting time, response time
- loss probability
- jitter

Operator performance
- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service
- contract between user and provider
- service guarantees
- tradeoff between utilization and QoS
Networking

Flow performance
- latency, waiting time, response time
- loss probability
- jitter

Operator performance
- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service
- contract between user and provider
- service guarantees
- tradeoff between utilization and QoS
Networking

Flow performance
- latency, waiting time, response time
- loss probability
- jitter

Operator performance
- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service
- contract between user and provider
- service guarantees
- tradeoff between utilization and QoS
Parallel processing

Program execution
- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization
- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling
- granularity of the application
- tradeoff between utilization and makespan
Parallel processing

Program execution
- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization
- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling
- granularity of the application
- tradeoff between utilization and makespan
Parallel processing

Program execution
- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization
- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling
- granularity of the application
- tradeoff between utilization and makespan
Distributed applications

**Application**
- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

**System utilization**
- service availability
- resource utilization
- communication throughput

**System security**
- reliability (error-free period)
- availability
Distributed applications

**Application**
- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

**System utilization**
- service availability
- resource utilization
- communication throughput

**System security**
- reliability (error-free period)
- availability
Distributed applications

Application
- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

System utilization
- service availability
- resource utilization
- communication throughput

System security
- reliability (error-free period)
- availability
**User point of view**

- optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

**Resource point of view**

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks

**Tradeoff Performance - Cost**
User point of view

optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

Resource point of view

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks
User point of view

optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

Resource point of view

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks

Tradeoff Performance - Cost
Why experiments?

Design of architectures, softwares

- System debugging (!!)
- Validation of a proposition
- Qualification of a system
- Dimensioning and tuning
- Comparison of systems

Many purposes $\Rightarrow$ different methodologies
Scientific Method

**Falsifiability** is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

Modeling comes before experimenting

**Modeling principles [J-Y LB]**

- (Occam:) if two models explain some observations equally well, the simplest one is preferable
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.
Scientific context

**Scientific Method**

**Falsifiability** is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

**Modeling comes before experimenting**

**Modeling principles [J-Y LB]**

- (Occam:) if two models explain some observations equally well, the simplest one is preferable.
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.
Scientific context

**Scientific Method**

**Falsifiability** is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

**Modeling comes before experimenting**

**Modeling principles [J-Y LB]**

- (Occam:) if two models explain some observations equally well, the simplest one is preferable
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.
Design of models (introduction)

Formulation of the question

Give explicitly the question (specify the context of experimentation)
- Identify parameters (controlled and uncontrolled)
- Identify factors (set levels)
- Specify the response of the experiment

Minimize the number of experiments for a maximum of accuracy
Ouvrages de référence orientés évaluation de performances

*Performance evaluation of computer and communication systems*, Jean-Yves Le Boudec, EPFL Press 2011

Ouvrages de référence orientés évaluation de performances

- The art of computer system performance analysis, Raj Jain, Wiley 1991
- Measuring Computer Performance, Lilja, Cambridge Univ Press 2005
- Simulation, Sheldon M. Ross, Wiley 2012