Observed Program Performance Variability

- Fixed binary program and a fixed machine with a fixed software environment with fixed data input.
- Repeat the program execution $n$ times.
- You get $n$ distinct performance numbers.
- If you do not observe variability, you do not need statistics.
Why program performances vary?

- Physical technology
  - Variable CPU frequency, asynchronous peripherals, input/output.
- Processor micro-architecture designs
  - OoO execution, branch prediction, data pre-fetching, memory hierarchy, etc.
- Competition between peripherals
  - Shared memory, NUMA, communication network.
- Operating systems
  - Virtual memory management, process memory layout, thread and process scheduling.
- Algorithmic factors
  - Parallel programming with imbalanced workload.
  - Non deterministic algorithms.
### Example of experiences on a dedicated machine

**gcc 312_swim_m 2 threads**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>No Affinity</th>
<th>With affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**gcc 312_swim_m 4 threads**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>No Affinity</th>
<th>With affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**gcc 312_swim_m 6 threads**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>No Affinity</th>
<th>With affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example of experience on a shared machine with dedicated CPU

**GNU Fortran Ifort**

Compiler = GNU Fortran (GCC) 4.4.6

Compiler = Ifort (IFORT) 12.1.0

Linux Kernel = 2.6.32

Input = stdin

Executions = 35

Intel Xeon E5 2670, 2.60 GHz, 16 cores, 64 GB RAM

### Before doing statistics: basic assumptions

- Independence of the measurements.
- Continuous model vs. discrete model.

### Parametric vs. non-parametric statistics

- **Parametric statistics**: assume a p.d.f.
  - Ex: Student t-test.
- **Non-Parametric statistics**: any kind of data distribution.
  - Ex: Wilcoxon-Mann-Whitney, chi-square test, central limit theorem.
Advantages of parametric statistics

- Parameters of a parametric model can often be interpreted.
- Formal mathematical proofs.
- More accurate for large data sets and multi-dimensional data.

Advantages of non-parametric statistics

- Do not need a mathematical model.
- Do not need to build new statistics for every data distribution.
- For large data sets, it works pretty well (but without proof).

Standard program performance metrics

- Summarise $n$ performance numbers by a single one:
  - Mean
  - Median
  - Minimum
  - Maximum
- You lose information.

How to model precisely the performances of a program to be able to take reliable conclusions/decisions?

Are the data distributions of a Gaussian nature?

- 10 years of collected data performances
- Various machines architectures and configurations.
- SPEC CPU applications (2001, 2006), all SPEC OMP applications, NAS Parallel Benchmark, own micro-benchmarks, other parallel applications, various compilers versions and options, Linux versions.
- 2438 data samples, each one contains between 30 and 1000 execution times.
- We did a normality test (Shapiro).
- With a risk of 5%, 67% of the samples are not of Gaussian nature.
Standard tests in statistics: Speedup-Test for program performances

- t-test of Student to compare between two theoretical means.
  - Program performances do not follow normal distributions in general.
- Wilcoxon-Mann-Whitney to compare between two theoretical medians.
  - Is a single median a good summary of n data?

Some known theoretical density functions

- Gaussian law
  - Used in physics

- Student law
  - Used to compute the confidence interval of the average

Which function can we use to model a continuous random variable?

Some facts

- Every sample has an infinite possible theoretical models.
- In the past, some density functions have been selected for their mathematical characteristics (to ease formal problem solving by hand).
- It is impossible to formally prove that one model is better than another, the theoretical distribution remains always unknown.
- Statistics do not provide guarantees, they help decision and analysis of complex/random phenomena.
Some known theoretical density functions

**Pareto law**

*Used in queue theory, quality management, etc.*

**Exponential law**

*Used in electronics and in radioactivity.*

**Gamma law**

*It is a generalisation of other laws.*

**Lognormal law**

*Used in finance, stock market, and many other domains.*
Parametric statistics for program performance analysis, comparison and evaluation

Some known theoretical density functions

Weibull law

Used in the science of materials

Many programs performances follow multi-modal distributions

Gaussian mixture

The program performances is modelled with a probability density function equal to the sum of $K$ gaussians:

$$ f_X(u) = \sum_{k=1}^{K} \pi_k \varphi_X(\mu_k, \sigma_k, u) $$

where $\sum_{k=1}^{K} \pi_k = 1$ and $\varphi_X(\mu_k, \sigma_k)$ is the probability density function of a gaussian with mean $\mu_k$ et standard deviation $\sigma_k$.

$$ \varphi_X(\mu, \sigma, u) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(u-\mu)^2}{2\sigma^2}} $$

Clustering in statistics

Given a data sample of size $n$, compute estimators of:

- The number $K$ of gaussians (called clusters);
- The weight $\pi_k$ of each cluster $k$;
- The theoretical mean $\mu_k$ of each cluster $k$;
- The standard deviation $\sigma_k$ of each cluster $k$.

A famous algorithm called $EM$ does the job.
**Examples of clustering**

![Graphs showing examples of clustering](image1.png)

**Checking the fitting of the data to the GM (Kolmogorov-Smirnov test)**

We designed a KS test adequately calibrated with bootstrap.

![Graphs showing the fitting test](image2.png)

**Experiments**

**Clustering results**

with a risk error of 5%

- 83% of the samples are modelled correctly with GM.
- 17% of the samples are rejected.

**Why 17% of the sample are rejected?**

- Ties (identical values), rounding errors when collecting performance data.
  - this problem is easily fixed by increasing the precision of the measurements.
- Heavy-tail distributions cannot be approximated easily with GM.
New performance metrics

Let $X$ and $Y$ be two random variables representing the performances of two programs.

1. The mean difference: $I_1 = \mathbb{E}[|X - Y|]$
2. The probability that a single program run is better than another: $I_2 = \mathbb{P}[X < Y]$.
3. The probability that a single run is better than all the others: $I_3 = \mathbb{P}[X_i < \min(X_2, \ldots, X_n)] = \mathbb{E}[\mathbb{1}_{X_i < \min(X_2, \ldots, X_n)}]$
4. The variability level $I_4$: the number of modes.

| Metric 1: $\mathbb{E}[|X - Y|]$ |
|---------------------------|
| $\mathcal{X}$ is a sample of size $n$, $\mathcal{Y}$ is a sample of size $m$. $X$ and $Y$ are approximated by GM $(K, \mu, \sigma)$ and $(K', \mu', \sigma')$ respectively. |
| 1. Non-parametric estimation: |
| $\mathbb{E}[|X - Y|] = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |x_i - y_j|$ |
| 2. Parametric estimation with GM: |
| $\mathbb{E}[|X - Y|] = \sum_{i=1}^{K} \sum_{j=1}^{K'} \mathbb{P}[\mathbb{1}_{X_i < \min(X_2, \ldots, X_n)}] \Phi\left(\frac{\mu_i - \mu'}{\sqrt{\sigma_i^2 + \sigma'^2}}\right) + \sqrt{2(\sigma_i^2 + \sigma'^2)} e^{-\frac{(\mu_i - \mu')^2}{2(\sigma_i^2 + \sigma'^2)}}$ |
| where: $\Phi$ is the CDF of the Gaussian $\phi(0, 1, u)$ |

<table>
<thead>
<tr>
<th>Metric 2: $\mathbb{P}[X &lt; Y]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{X}$ is a sample of size $n$, $\mathcal{Y}$ is a sample of size $m$. $X$ and $Y$ are approximated by GM $(K, \mu, \sigma)$ and $(K', \mu', \sigma')$ respectively.</td>
</tr>
<tr>
<td>1. Non-parametric estimation (Man-Whitney statistic):</td>
</tr>
<tr>
<td>$\mathbb{P}[X &lt; Y] = \mathbb{E}[\mathbb{1}<em>{X &lt; Y}] = \frac{1}{mn} \sum</em>{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{X_i &lt; Y_j}$</td>
</tr>
<tr>
<td>2. Parametric estimation with GM:</td>
</tr>
<tr>
<td>$\mathbb{P}[X &lt; Y] = \sum_{i=1}^{K} \sum_{j=1}^{K'} \mathbb{P}[\mathbb{1}_{X_i &lt; \min(X_2, \ldots, X_n)}] \Phi\left(\frac{\mu_i - \mu'}{\sqrt{\sigma_i^2 + \sigma'^2}}\right)$</td>
</tr>
<tr>
<td>where: $\Phi$ is the CDF of the Gaussian $\phi(0, 1, u)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric 3: $\mathbb{P}[X_1 \leq \min(X_2, \ldots, X_m)]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{X}$ is a sample of size $n$, $\mathcal{Y}$ is a sample of size $m$. $X$ and $Y$ are approximated by GM $(K, \mu, \sigma)$ and $(K', \mu', \sigma')$ respectively.</td>
</tr>
<tr>
<td>1. Non-parametric estimation:</td>
</tr>
<tr>
<td>$\mathbb{P}[X_1 \leq \min(X_2, \ldots, X_m)] = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{X_i &lt; \min(X_2, \ldots, X_n)}$</td>
</tr>
<tr>
<td>2. Parametric estimation. We note $Y = \min(X_2, \ldots, X_m)$ and $G$ its CDF. $\mathbb{P}[X &lt; Y] = \mathbb{E}[1 - G(x)] = \int (1 - G(u)) f_1(u) du$</td>
</tr>
<tr>
<td>where $f_1(u) = \sum_{i=1}^{K} \Phi\left(\frac{\mu_i - \mu}{\sqrt{\sigma_i^2 + \sigma'^2}}\right)$</td>
</tr>
</tbody>
</table>
Metric 4: Variability level

- The variance is a measure of dispersion around the average, not easy to interpret!
  \[ P[|X - \mu| > 4\sigma_X] \] is very small
- Our proposal: variability level = number of modes.

Empirical study of variability levels of programs execution times:

- The variability level: the number of modes.
  - \( \approx 37\% \) of the samples have a variability level equal to 1.
  - \( \approx 32\% \) of the samples have a variability level equal to 2.
  - \( \approx 12\% \) of the samples have a variability level equal to 3.
  - \( \approx 19\% \) of the samples have a variability level \( \geq 4 \).

Limitations

- Some cases (Heavy-tail distributions) cannot be well modelled with GM.
- GM are not appropriate models for studying extreme value performances (minimum and maximum).

Future plan

- Consider multi-dimensional performance data: tuples of values of distinct nature \{execution time, energy consumption, memory consumption, network traffic, etc\}.
- It is very difficult to have satisfactory regression models for multi-dimensional data.
- GM models are well adapted for such mathematical modelling.
Conclusions

- Observed program performances are multi-modal;
- Modelling program performances with gaussian mixtures;
- Test of the fitting between the GM model and the data;
- New performance metrics;
- Free software called VARCORE.

Main reference

Full research report (70 pages) with free software and demo. 
*Parametric and Non-Parametric Statistics for Program Performance Analysis and Comparison.*
https://hal.inria.fr/hal-01286112

VARCORE software

- Programmed with R.
- Requires free packages: mclust, R.utils
- Documented.

Example 1: analysing the performances of one program

- Load performance data of one program
- Apply clustering (build a GM model)
- Check if the GM model is good enough.
- Compute the variability level.
- Print and plot the results.
Example 2: comparing the performances of multiple codes versions

- Load performance data of multiple codes versions.
- Apply clustering (build a GM model) for each one.
- Check if the GM model is good enough for each one.
- Decide which is the best code version.
- Compute other performance metrics.