Extra-P - Performance Modeling at Scale

Marcus Ritter¹, Alexander Geiß¹, Gustavo Morais¹, Alexandru Calotoiu², Torsten Hoefler², and Felix Wolf¹

¹ Technical University of Darmstadt, ² ETH Zürich
Spectrum of performance analysis methods
Scaling your code can harbor *performance surprises*…

*Goldsmith et al., 2007*
Performance model

Formula that expresses relevant performance metric as a function of one or more execution parameters

\[ t = 3 \cdot 10^{-4} p^2 + c \]

Analytical (i.e., manual) creation challenging for entire programs

- Identify kernels
- Create models
- Incomplete coverage
- Laborious, difficult
Analytical performance modeling

- **Identify kernels**
  - Parts of the program that dominate its performance at larger scales
  - Identified via small-scale tests and intuition

- **Create models**
  - Laborious process
  - Still confined to a small community of skilled experts

**Disadvantages:**

- Time consuming
- Danger of overlooking unscalable code

---


Bauer et al.: *Analysis of the MILC Lattice QCD Application su3_rmd.* CCGrid, 2012
Automatic empirical performance modeling with Extra-P

main() {
foo()
bar()
compute()
}

Instrumentation
Performance measurements (profiles)

All functions

Input

Output

• Human-readable performance models (e.g. f(p) = 50 + 3log(p))
• Color coded ranking

Primary focus on scaling trend

Common performance analysis chart in a paper

Ranking

1. $F_2$
2. $F_1$
3. $F_3$
Primary focus on scaling trend

Actual measurement in laboratory conditions

Ranking

1. $F_2$
2. $F_1$
3. $F_3$
Primary focus on scaling trend

Production Reality

Ranking

1. $F_2$
2. $F_1$
3. $F_3$
Model building blocks

**Computation**
- LU: $t(p) \sim c$
- FFT: $t(p) \sim \log_2(p)$
- Naïve N-body: $t(p) \sim p$
- Samplesort: $t(p) \sim p^2 \log_2(p)$

**Communication**
- LU: $t(p) \sim c$
- FFT: $t(p) \sim \log_2(p)$
- Naïve N-body: $t(p) \sim p$
- Samplesort: $t(p) \sim p^2$
Performance model normal form

\[ f(p) = \sum_{k=1}^{n} c_k \times p^{i_k} \times \log_{2}^{j_k}(p) \]

\[ n = 1 \]
\[ I = \{0, 1, 2\} \]
\[ J = \{0, 1\} \]
Performance model normal form

\[ f(p) = \begin{cases} 
  c_1 \times \log(p) + c_2 \times p \\
  c_1 \times \log(p) + c_2 \times p \times \log(p) \\
  c_1 \times \log(p) + c_2 \times p^2 \\
  c_1 \times \log(p) + c_2 \times p \times \log(p) \\
  c_1 \times \log(p) + c_2 \times p^2 \times \log(p) \\
  c_1 \times \log(p) + c_2 \times p \times \log(p) \\
  c_1 \times \log(p) + c_2 \times p^2 \times \log(p) \\
  c_1 \times \log(p) + c_2 \times p^2 \times \log(p) \\
  c_1 \times p + c_2 \times p \times \log(p) \\
  c_1 \times p + c_2 \times p^2 \times \log(p) \\
  c_1 \times p^2 + c_2 \times p \times \log(p) \\
  c_1 \times p^2 + c_2 \times p^2 \times \log(p) \\
  c_1 \times p^2 + c_2 \times p^2 \times \log(p) \\
\end{cases} \]

\( n = 1 \)
\( I = \{0, 1, 2\} \)
\( J = \{0, 1\} \)
Workflow

Statistical quality control

Performance measurements

Performance profiles

Model generation

Model refinement

Scaling models

Accuracy saturated?

Yes

No

Kernel refinement

Scaling models

Performance extrapolation

Ranking of kernels
Assumptions & limitations

- Only one scaling behavior for all the measurements; no jumps
- Some MPI collective operations switch their algorithm – results in bad models

Example: red model tries to model measurements of different algorithms

- First 4 points – one function
- Last 4 points – another function (linear)
- Adj. R2 = 0.95085 (!)
Changing growth trends

• Ranking according to growth rate difficult:

• $\log^2(p)$ vs. $\sqrt{p}$
Changing growth trends (2)
Ranking of kernels

- Kernels are ranked according the leading-order terms in the models

- Leading-order term $\rightarrow$ big-O notation

- For example: $O(x)$ comes before $O(x^2)$
Performance measurements

- Different ways of collecting measurements
- Score-P (http://www.vi-hps.org/projects/score-p/)
- Other profiling tools, e.g. HPCToolkit
- Manual ad-hoc measurements
Performance measurements (2)

- At least 5 different measurements required

Performance measurements (profiles)

\[ p_1 = 256 \]
\[ p_2 = 512 \]
\[ p_3 = 1024 \]
\[ p_4 = 2048 \]
\[ p_5 = 4096 \]
Performance measurements (3)

- At least 5 different measurements required

Each measurement repeated multiple times

Performance measurements (profiles)

\[ p_1 = 256 \]
\[ p_2 = 512 \]
\[ p_3 = 1024 \]
\[ p_4 = 2048 \]
\[ p_5 = 4096 \]

... 

Noisy results
Variance too big

Processes
Time

2^8 2^9 2^{10} 2^{11} 2^{12}
Statistical quality control

• If the **confidence interval** is too wide, the fit will not be optimal, or overfitting might occur

\[ CI = f(\text{mean, stddev}) \]

• To improve CI - increase repetitions, include different configurations

---

Performance metric

- **Unknown behavior of kernel**
- Data(with noise)
- Confidence interval (t-Test)

Confidence band – noise uncertainty

12/5/2022 | Department of Computer Science | Laboratory for Parallel Programming | Marcus Ritter | 21
How to select the best model? - adjusted $R^2$

- $R^2$ represents how well the determined function fits the $M$ available measurements.
- Adjusted $R^2$ adjusts for $N$, the number of terms used.
- Adj. $R^2$ decreases $\rightarrow$ more useless variables.
- Adj. $R^2$ increases $\rightarrow$ more useful variables.
- Rule of thumb: adj. $R^2 > 0.95$

\[
R^2 = 1 - \frac{\text{residual SumSquares}}{\text{total SumSquares}}
\]

\[
\bar{R^2} = 1 - (1 - R^2) \times \frac{M}{M - N - 2}
\]
Extra-P 3.0: Fast multi-parameter performance modeling

Consider the performance difference:

\[ f(p, n, o, \epsilon) = 10 + \sqrt{p} + 10 \times n + o^2 \]

\[ f(p, n, o, \epsilon) = 10 + p \times 10 \times n \times o^2 \]
Extra-P 3.0: Fast multi-parameter performance modeling

Expanded performance model normal form

\[ f(p) = \sum_{k=1}^{n} \sum_{l=1}^{m} c_k p_{i_l}^{j_{l_k}} \times \log_2 (p_l) \]

Model candidates

- Constant: \( c_1 \)
- Single parameter: \( c_1 + c_2 \times p_1 \)
- Multiple parameters: 
  - Additive: \( c_1 + c_2 \times p_1 + c_3 \times p_2 \)
  - Multiplicative: \( c_1 + c_2 \times p_1 \times p_2 \)
  - Complex: \( c_1 + c_2 \times p_1 + c_3 \times p_1^2 \times p_2 \times \log_2 (p_2) \)
Extra-P 4.0: Sparse modeling

- Experiments can be expensive
- So far we needed $5^{(m+1)}$ experiments, $m=$number of parameters
Extra-P 4.0: Sparse modeling

- Using our new sparse modeling approach we can model with less points!
- We only need $5^m$ experiments, $m=$number of parameters
Extra-P 4.0: Sparse modeling

- Experiment configuration strategy using our heuristic guideline

Measure min. amount points required for modeling

Create a model using Extra-P

Cost < Budget?

- yes
  - Gather on additional measurement
- no
  - Evaluate against previous model

Create new model

Final model

Cost < Budget?

- yes
  - Gather on additional measurement
- no
  - Evaluate against previous model

Final model
Extra-P 4.0: Sparse modeling

• Using sparse modeling we can reduce the average modeling cost by ~85% (on synthetic data)
• We can retain ~92% of the model accuracy (on synthetic data)
• Allows a more flexible experiment design
• **FASTEST**: 70% decrease in cost, ~2% prediction error
• **Kripke**: 99% decrease in cost, ~39% prediction error
• **Relearn**: 85% decrease in cost, ~11% prediction error
Using Extra-P 4.0
Extra-P requirements

- Python 3.7 or higher
- numpy
- pycubexr
- marshmallow
- tqdm
- PySide2 (for GUI)
- matplotlib (for GUI)
- pyobjc-framework-Cocoa (only for GUI on macOS)
Installing Extra-P

• Easy to install via pip

• Just run: python -m pip install extrap --upgrade

• The --upgrade forces the installation of a new version

• All dependencies (packages) will be installed automatically
Extra-P in the tuning workshop

- Install Extra-P either on your own computer or on the cluster

- Example input data provided at: https://gigamove.rwth-aachen.de/en/download/4e812f0fe2e5a2134aa2b2af398f2d3a

- We will show you how to use Extra-P using the example data sets or your own

- Usage of the command line tool

- How to utilize the created performance models to obtain further insights
Extra-P in the tuning workshop

• Available at: https://github.com/extra-p/extrap

• When installed on the system simply run:
  ▪ `extrap` – for the command line version
  ▪ `extrap-gui` – for the graphical user interface version

• The GUI version is not intended to be used on the cluster
Modeling Sets of Cube Experiments
Automatic empirical performance modeling with Extra-P

main() {
  foo()
  bar()
  compute()
}

Performance measurements (profiles)

- $p_1 = 32$
- $p_2 = 64$
- $p_3 = 128$
- $p_4 = 256$
- $p_5 = 512$
- $p_6 = 1024$

Instrumentation

All functions

Input

Output

- Human-readable performance models (e.g. $f(p) = 50 + 3\log_2(p)$)
- Color coded ranking

Extra-P

• Automatic empirical performance modeling with Extra-P
Extra-P Cube input description

- Modeling tool expects Cube files in the following format:
  - `<DIR>/<PREFIX><X><POSTFIX>.r<1,..,REPS>/<FILENAME>
  - `DIR`, `PREFIX`, `X`, `POSTFIX`, `REPS` and `FILENAME` must all be defined
    - `X` – value of varied parameter e.g. number of processes
    - `REPS` – number of repeated experiments with same parameter value
Extra-P Cube input description

<DIR>/<PREFIX><X><POSTFIX>.r<\{1,..,REPS}\>/<FILENAME>

<table>
<thead>
<tr>
<th>Open set of CUBE files</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open experiment</td>
<td>Ctrl+O</td>
</tr>
<tr>
<td>Save experiment</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td>Open text input</td>
<td></td>
</tr>
<tr>
<td>Screenshot</td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td>Ctrl+Q</td>
</tr>
</tbody>
</table>

Import Settings

Scaling type: weak

OK  Cancel
Visualization with Extra-P
Extra-P user interface
Extra-P call tree view

- Metric selection
- Model selection
- Call tree exploration
- Model
- Quality of fit metrics: Residual sum of squares and Adjusted $R^2$
- Impact of each kernel on the metric at the selected process count compared to the other kernels
- Asymptotic behavior
Extra-P model view

Models selected in the Call path view

Measurement values

X axis scale control for prediction of behavior at other process counts
Modeling Measurements from a Text File
## Choose input file

<table>
<thead>
<tr>
<th>Option</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open set of CUBE files</td>
<td></td>
</tr>
<tr>
<td>Open experiment</td>
<td>Ctrl+O</td>
</tr>
<tr>
<td>Save experiment</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td><strong>Open text input</strong></td>
<td></td>
</tr>
<tr>
<td>Screenshot</td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td>Ctrl+Q</td>
</tr>
</tbody>
</table>

Select input file in the GUI
Extra-P input in text form

- Useful when no CUBE files are available or when a small data set must be modeled

```
PARAMETER p
POINTS 1000 2000 4000 8000 16000
METRIC metric1
REGION region1
DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99
DATA 256.01 255.99 256 256
```

Parameter name
This name will be used in the GUI as well as in the textual output
PARAMETER p

POINTS 1000 2000 4000 8000 16000

METRIC metric1

REGION region1

DATA 1 1 1 1 1

DATA 4 4 4 3.99 4.01

DATA 16 15.999 16.01 16.01 15.99

DATA 64 64 64 64.01 63.99

Measurement points
Use at least 5, preferably 6, but in general the more the better
Extra-P input in text form

PARAMETER p
POINTS 1000 2000 4000 8000 16000
METRIC metric1
REGION region1
DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99

Metric name
Region name
Both used to determine the output Cube file hierarchical structure and identify separate data sets
Extra-P input in text form

PARAMETER p

POINTS 1000 2000 4000 8000 16000

METRIC metric1

REGION region1

DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99

Data points
Each row corresponds to a point; all values in a row are considered repeat measurements of the same experiment
Extra-P input in text form

PARAMETER p

POINTS 1000 2000 4000 8000 16000
METRIC metric1
REGION region1
DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99
DATA 256 256.01 255.99 256 256

Data points
Each row corresponds to a point; all values in a row are considered repeat measurements of the same experiment.
Utilizing the Command Line Tool
Extra-P command line tool

- Provides the same functionality, without visualization for use on cluster
- Usage guideline and command can be found at: https://github.com/extra-p/extrap

1.) Run: extrap

Command Format: extrap OPTIONS (--cube | --text | --talpas | --json | --extra-p-3) FILEPATH

2.) Select input type: extrap --text /input.txt
Extra-P command line tool

3.) Output:

Callpath: compute
Metric: time

Callpath, kernel of the application that was measured
Metric name; either Score-P metrics (time, bytes, etc.) or custom metrics

Measurements for each input element (e.g., #processes)

Best-fit model

Model: $-0.8897934098062804 + 0.20168243826499183 \times x^{(2)}$

RSS: 3.43E+01
Adjusted R^2: 1.00E+00

RSS: Residual sum of squares
Adjusted R^2 (explained previously)
Feedback

• What additional features would you like to see?

• Did you find any bugs?

• You can contact us via email: extra-p-support@lists.parallel.informatik.tu-darmstadt.de

• Or on GitHub using the issues tool: https://github.com/extra-p/extrap
Questions