# Tracing Performance Tools Back 25 Years

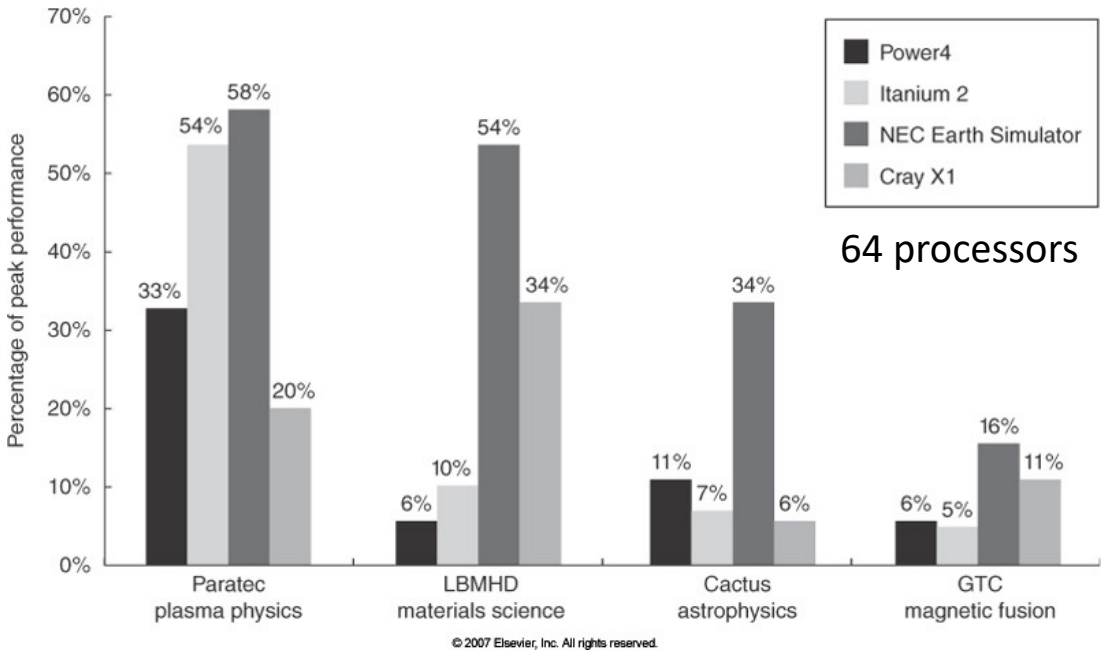
**Felix Wolf, Technical University of Darmstadt**

**25th Anniversary of APART**



Photo: Alex Becker / TU Darmstadt

# Starting point – peak performance gap



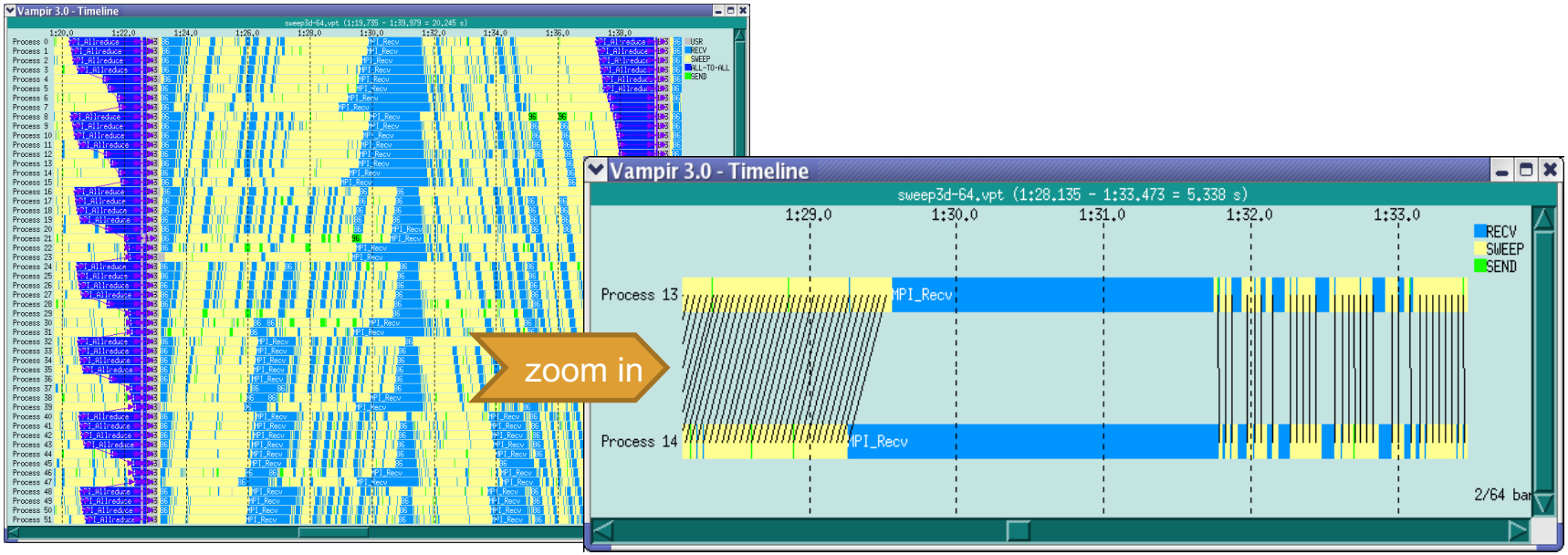

64 processors

Source: Hennessy, Patterson: Computer Architecture, 4th edition, Morgan Kaufmann

Peak performance is the performance a computer is guaranteed not to exceed…

# Event traces - too large to analyze manually

- Needles-in-the-hay-stack problem
- Low abstraction level - programmatic access needed




https://www.vampir.eu

# EARL - Event Analysis and Recognition Language

Extended Python interpreter

- Also Tcl, Perl interfaces
- Pattern specified in extended scripting language

Services

- Efficient random access to events
  - VAMPIR, ALOG, CLOG trace data formats
- Mapping of events to EARL event trace model

Abstractions defined in the EARL event trace model

- Permit use of simple pattern search algorithm

# EARL - Event Trace Model

## Basic Model

- Sequence of events: E = $e_1$,...,$e_n$
- Event types: enter, exit, send, recv
- Event attributes: time, loc, region, src, dest

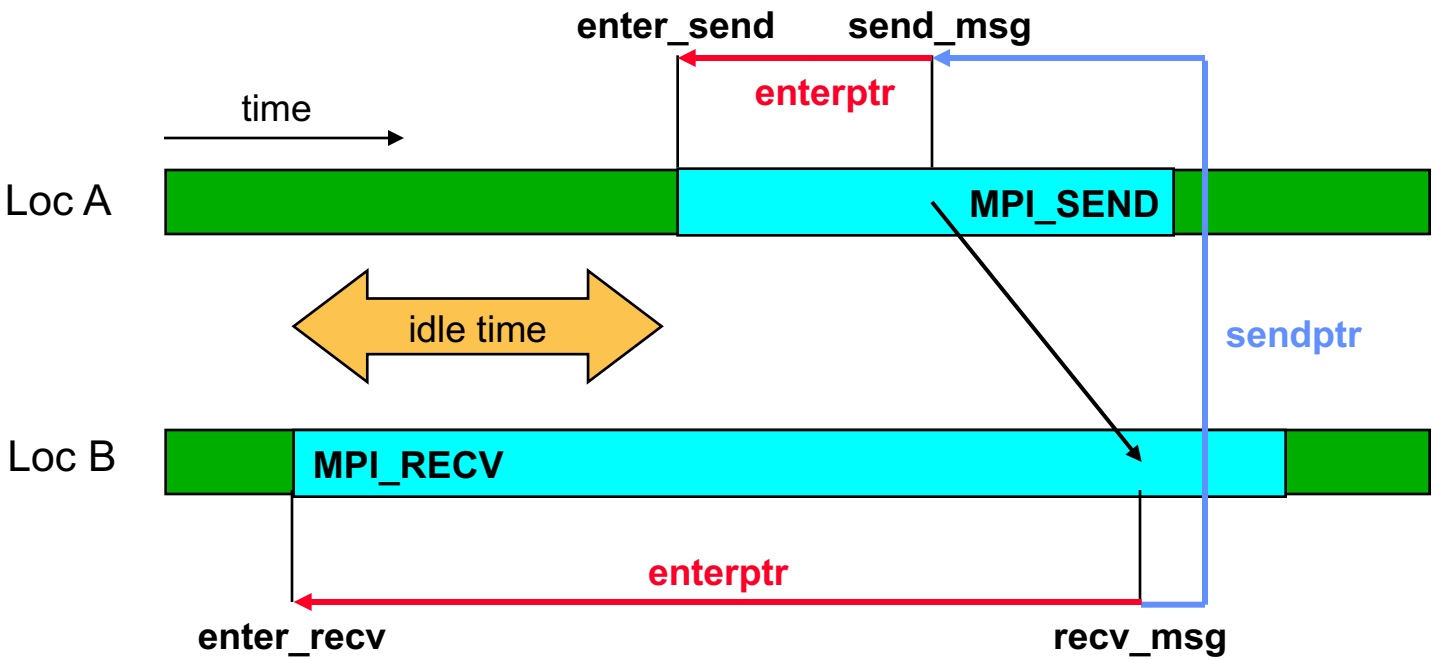
## Abstractions

- Pointer attributes connecting related events
  - enterptr : links event to entering of enclosing region instance
  - sendptr : links message receipt to message dispatch
- System states mapping events to execution context (set of events)
  - Region stack: set of enter events belonging to open region instances
  - Message queue: set of send events belonging to messages in transfer

# Example: Late Sender

MPI_RECV is posted before corresponding MPI_SEND

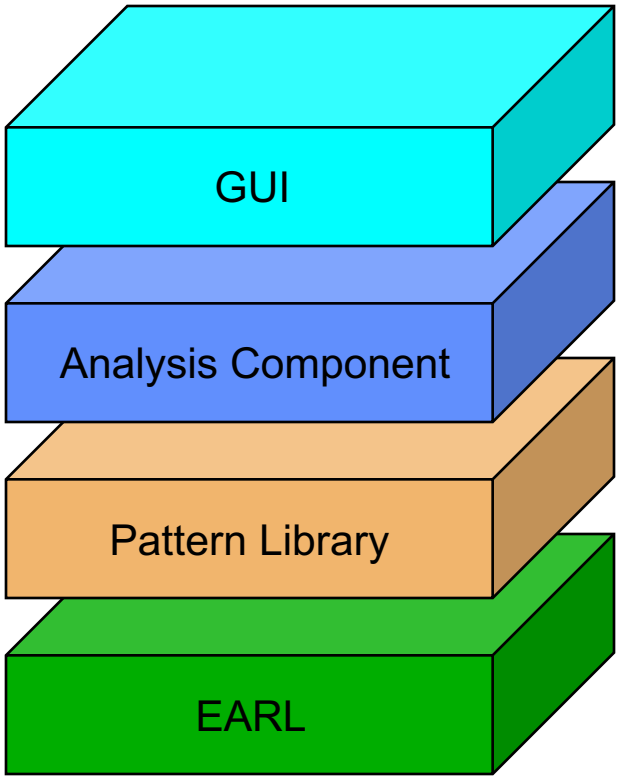
- Determine time during which receiver sits idle



# Pattern Class: LateSender

```
class LateSender(Pattern):
  [...]
  def recv_callback(self, recv_msg):
    enter_recv = self.trace.event(recv_msg['enterptr'])
    send_msg   = self.trace.event(recv_msg['sendptr'])
    enter_send = self.trace.event(send_msg['enterptr'])
    idle_time = enter_send['time'] - enter_recv['time']
    if (idle_time > 0  and
        enter_send['region'] == "MPI_SEND" and
        enter_recv['region'] == "MPI_RECV"):
      self.sum_idle_time = self.sum_idle_time + idle_time
  [...]
  def confidence(self): return 1
  def severity(self): return self.sum_idle_time
```

# EXPERT - Extensible Performance Tool



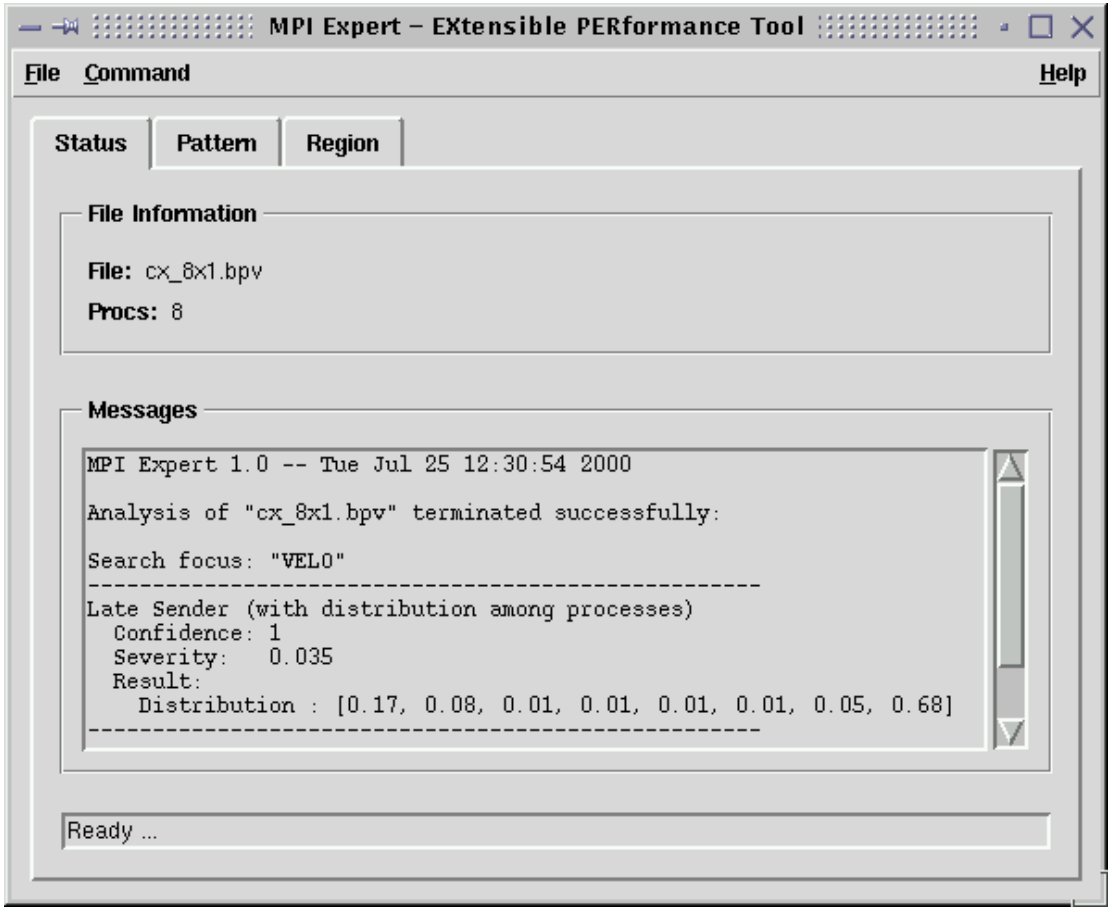

**User selection**
- Performance properties
- Code region (e.g. function)

**Pattern matching**
- Walking through trace file

**Separate module**
- Extensible by plug-ins

**Abstraction layer over event trace**

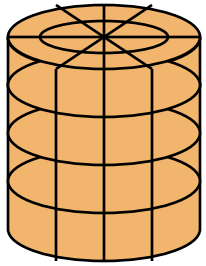
# EXPERT Output



# Test Case: Czochralski Crystal Growth

Convection processes in a rotating cylindric crucible

- 3 dimensional cubical mesh
- 2 dimensional spatial decomposition

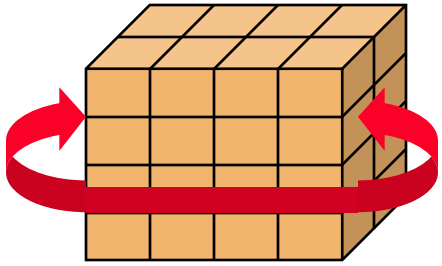


Most work is done in routine VELO

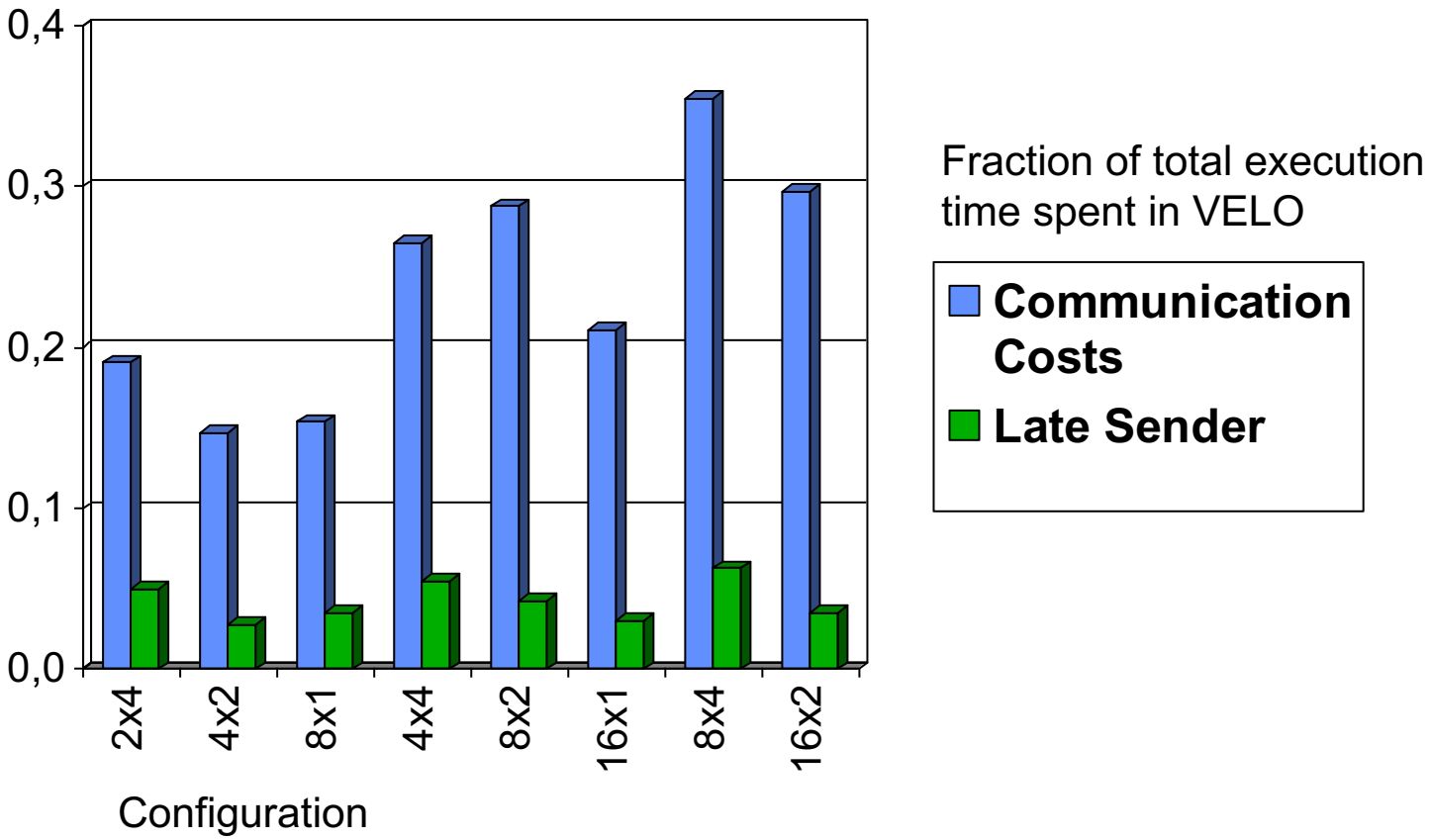
- Calculating new velocity vectors

Patterns used for analysis

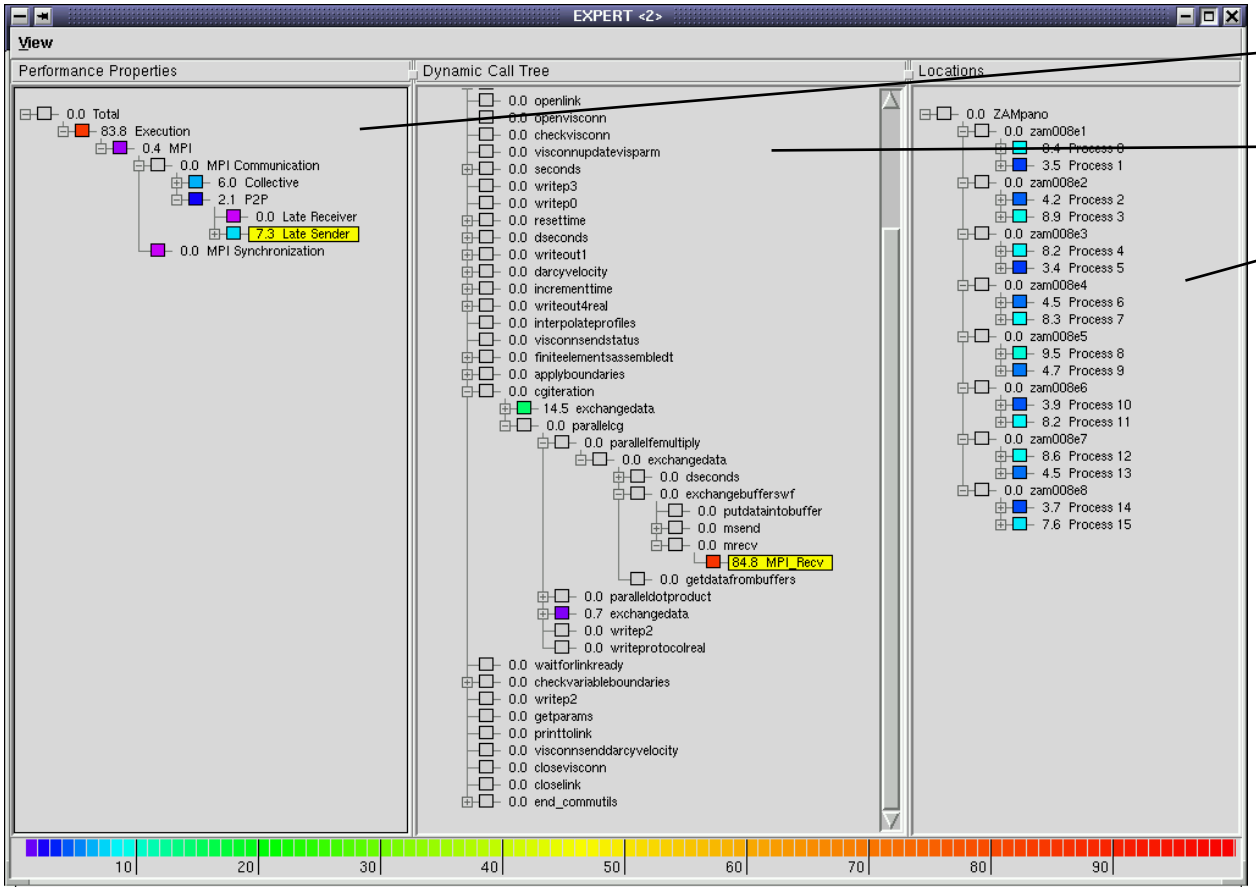
- Communication costs
- Late sender



# Idle times in VELO caused by Late Sender



# Next generation GUI



# History

- 1998
  - Start at the ZAM (Michael Gerndt and Bernd Mohr)
  - Manifesto

    M. Gerndt, B. Mohr, M. Pantano, F. Wolf: Automatic Performance Analysis for Cray T3E, Proceedings of the 7th Workshop on Compilers for Parallel Computers (CPC 98), University of Linköping, Sweden, June-July 1998

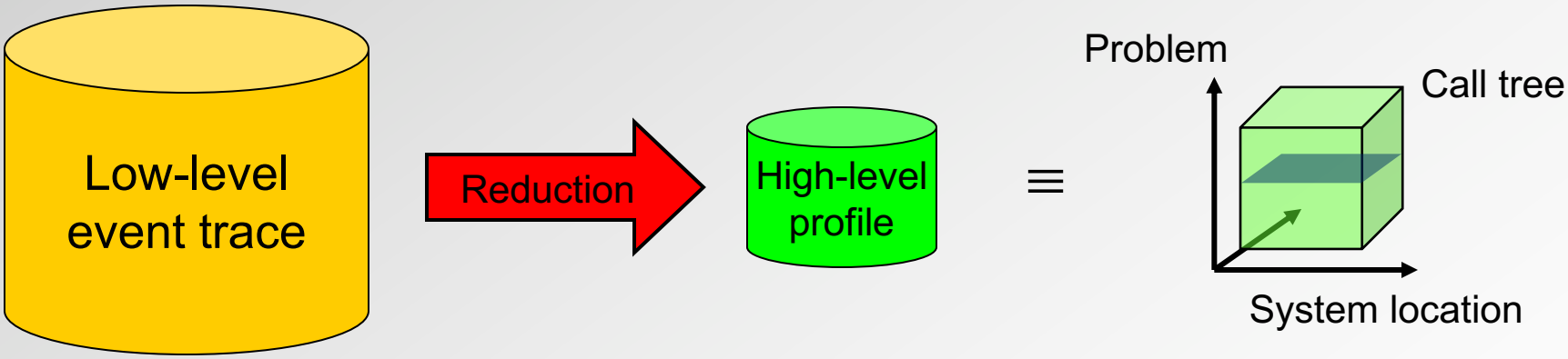
  - First component: EARL trace-analysis toolkit (Diploma thesis, ZAM)
- 2000
  - First prototype of an automatic trace analyzer (EXPERT)
- 2001
  - Automatic OpenMP instrumentation (OPARI, POMP)
  - First prototype of the EPILOG tracing library
  - First prototype of the KOJAK GUI

# History (2)

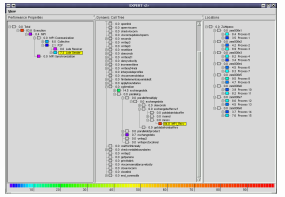
- 2003
  - KOJAK became collaboration between Forschungszentrum Jülich and the University of Tennessee
  - First beta release of KOJAK
- 2004
  - Release 1.0 and 2.0b
  - New GUI component CUBE
- 6 Diploma theses
- 1 Ph.D. thesis
- 22 publications
  - 3 journal articles
  - 19 conference and workshop articles

# Automatic performance analysis

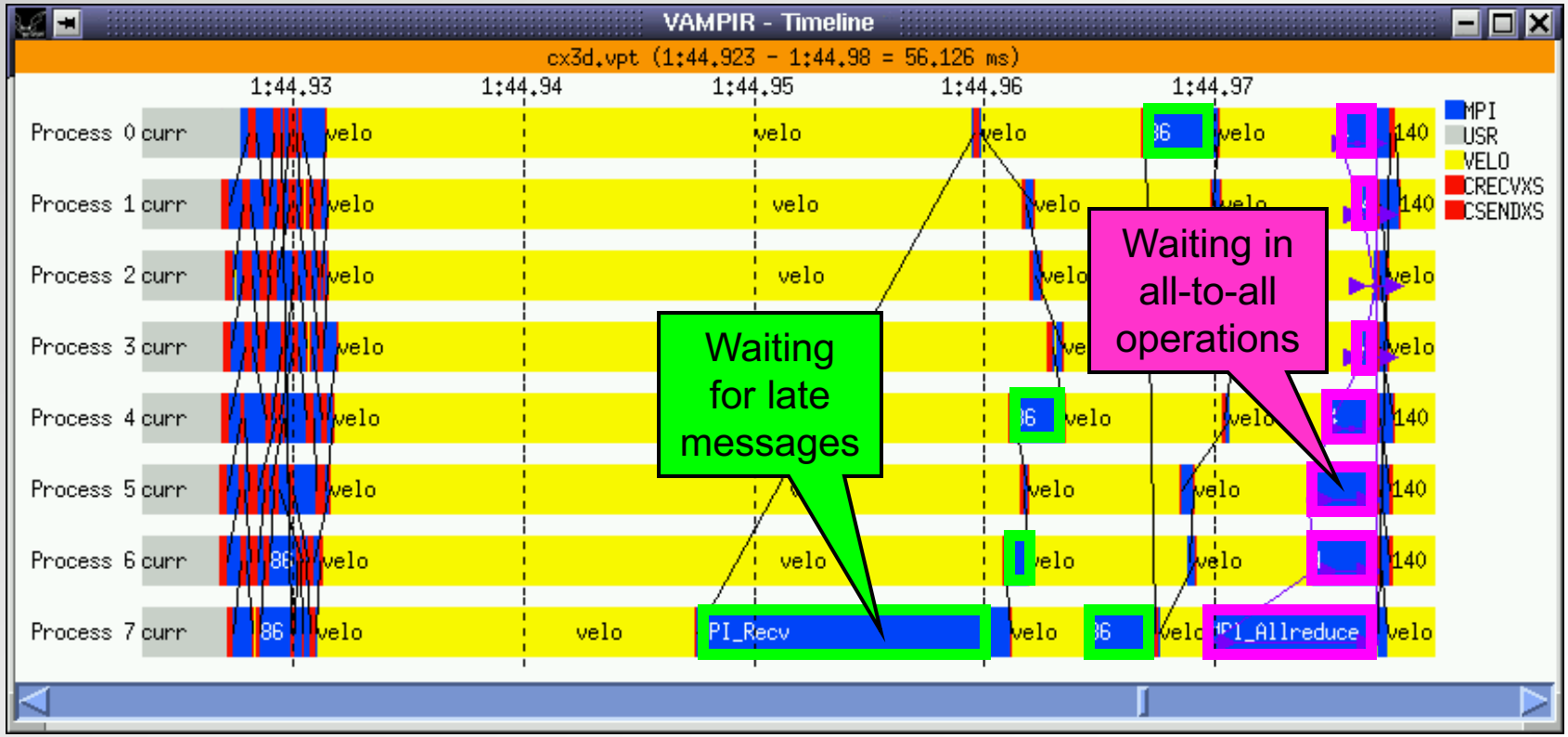
- Automatic performance analysis



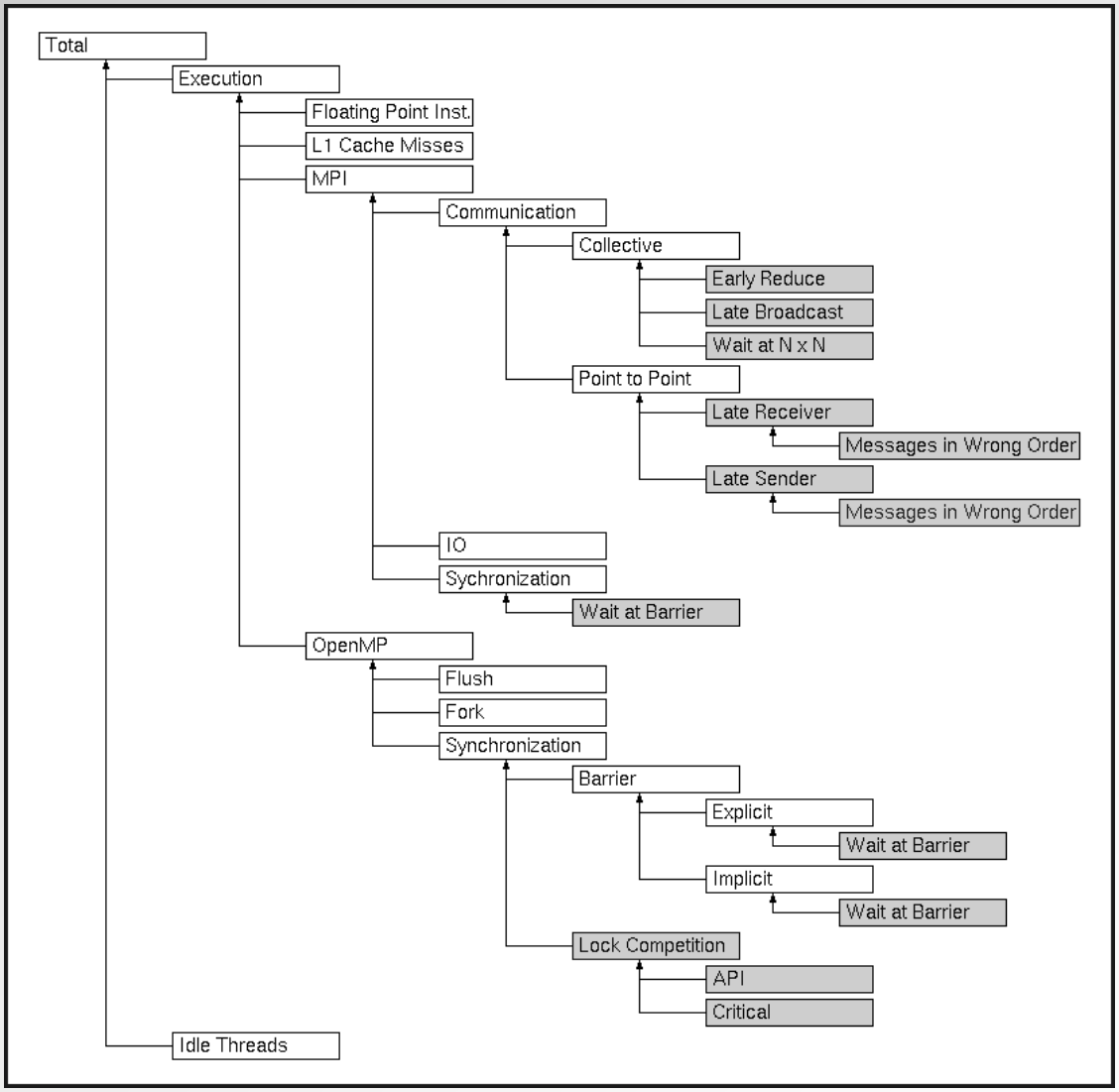

- Take event traces of MPI/OpenMP applications
- Search for execution patterns
- Calculate high-level call path profile
  - Problem, call path, system location ⇒ time
- Display in performance browser



# Typical performance problems

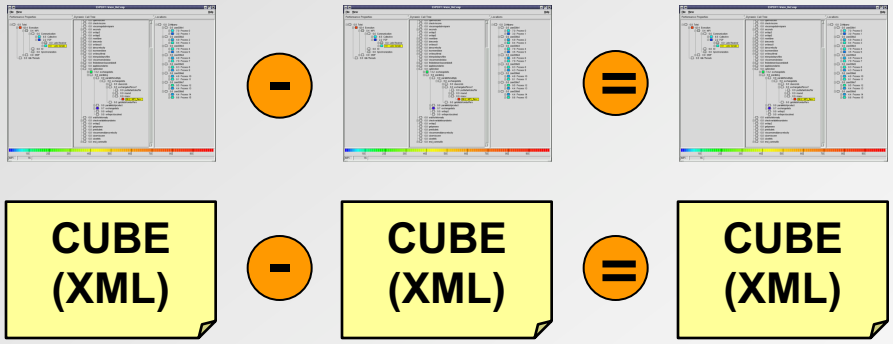


# Pattern hierarchy



# CUBE Performance algebra

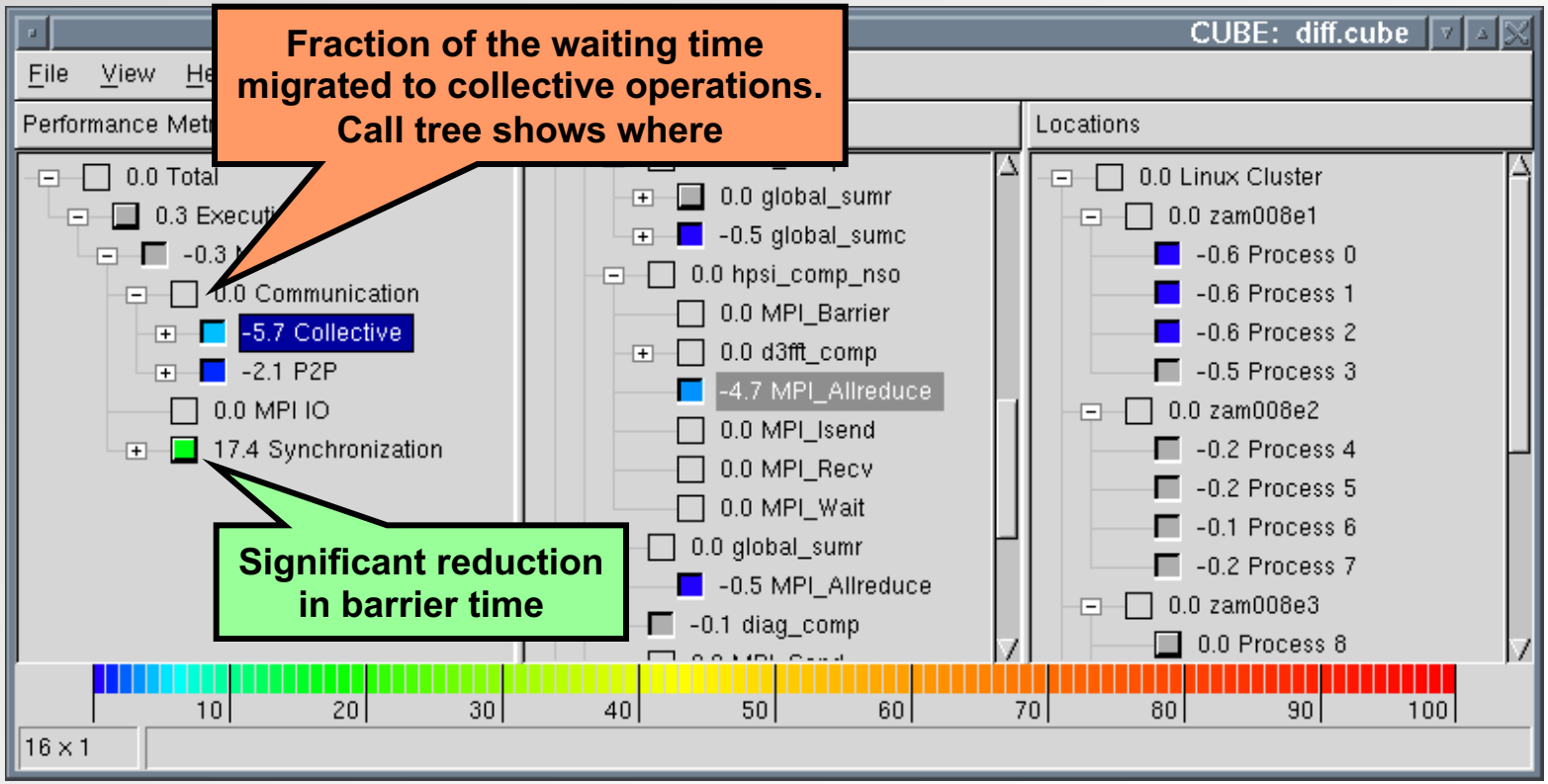
- Difference operator
  - Compare different experiments
- Merge operator
  - Integrate performance data from multiple sources
- Mean operator
  - Summarize a series of experiments




- Obtain new CUBE instance as result
- Display it like ordinary CUBE instance

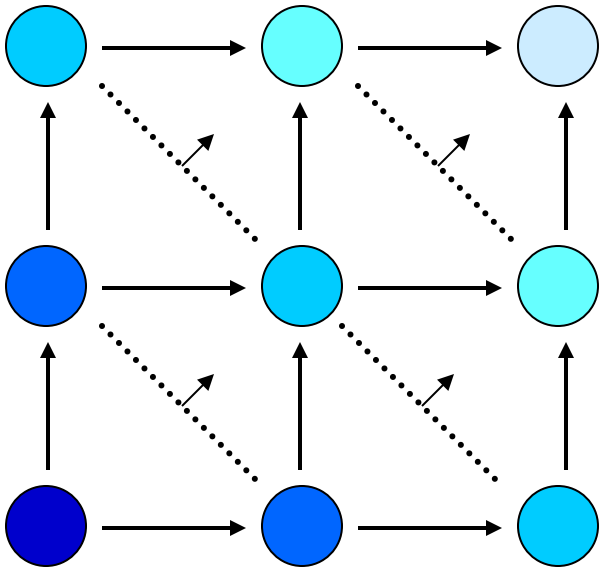
# (Re)moving waiting times

- Difference between before / after barrier removal
- Raised relief shows improvement
- Sunken relief shows degradation



# Virtual topology in SWEEP3D

- Three-dimensional domain (i,j,k)
- Two-dimensional domain decomposition (i,j)



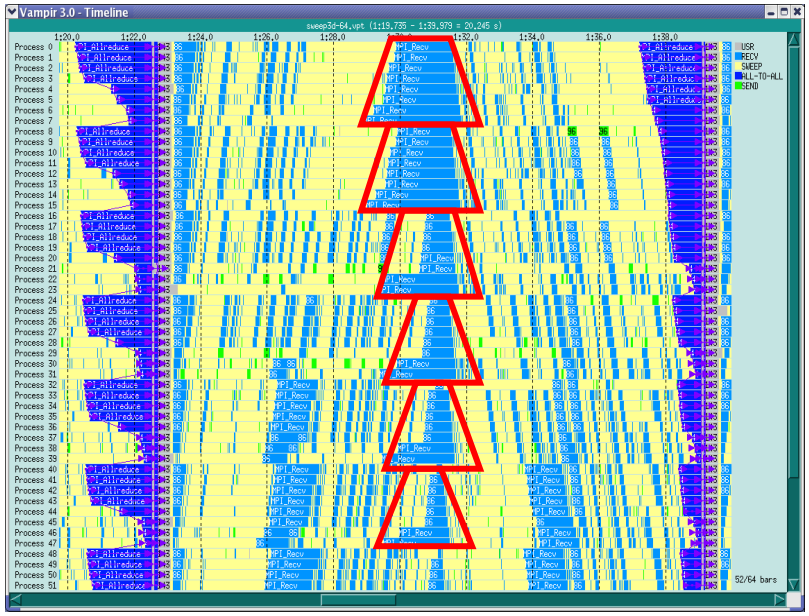
```
DO octants
  DO angles in octant
    DO k planes
      ! block i-inflows
      IF neighbor(E/W) MPI_RECV(E/W)
      ! block j-inflows
      IF neighbor(N/S) MPI_RECV(N/S)
         ... compute grid cell ...
      ! block i-outflows
      IF neighbor(E/W) MPI_SEND(E/W)
      ! block j-outflows
      IF neighbor(N/S) MPI_SEND(N/S)
    END DO k planes
  END DO angles in octant
END DO octants
```

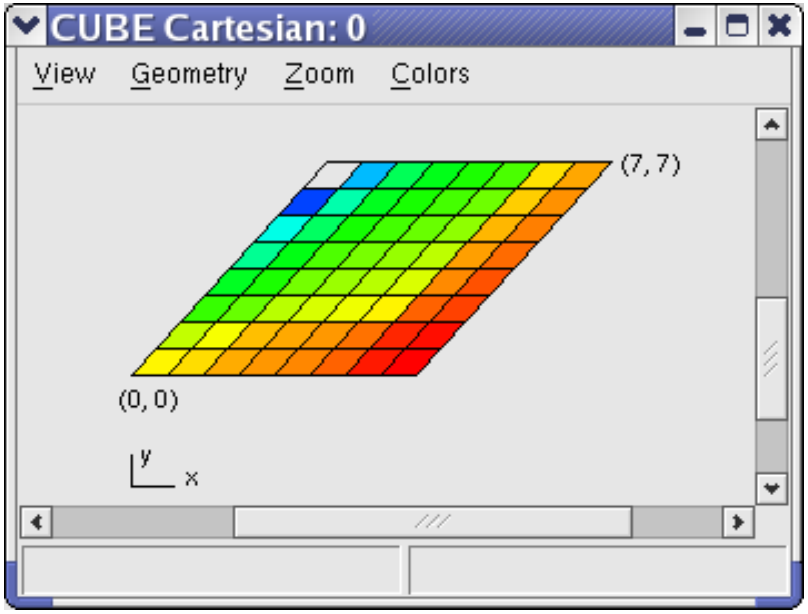
- Wave-fronts from different directions
- Limited parallelism upon pipeline refill (late sender)

# Four new patterns

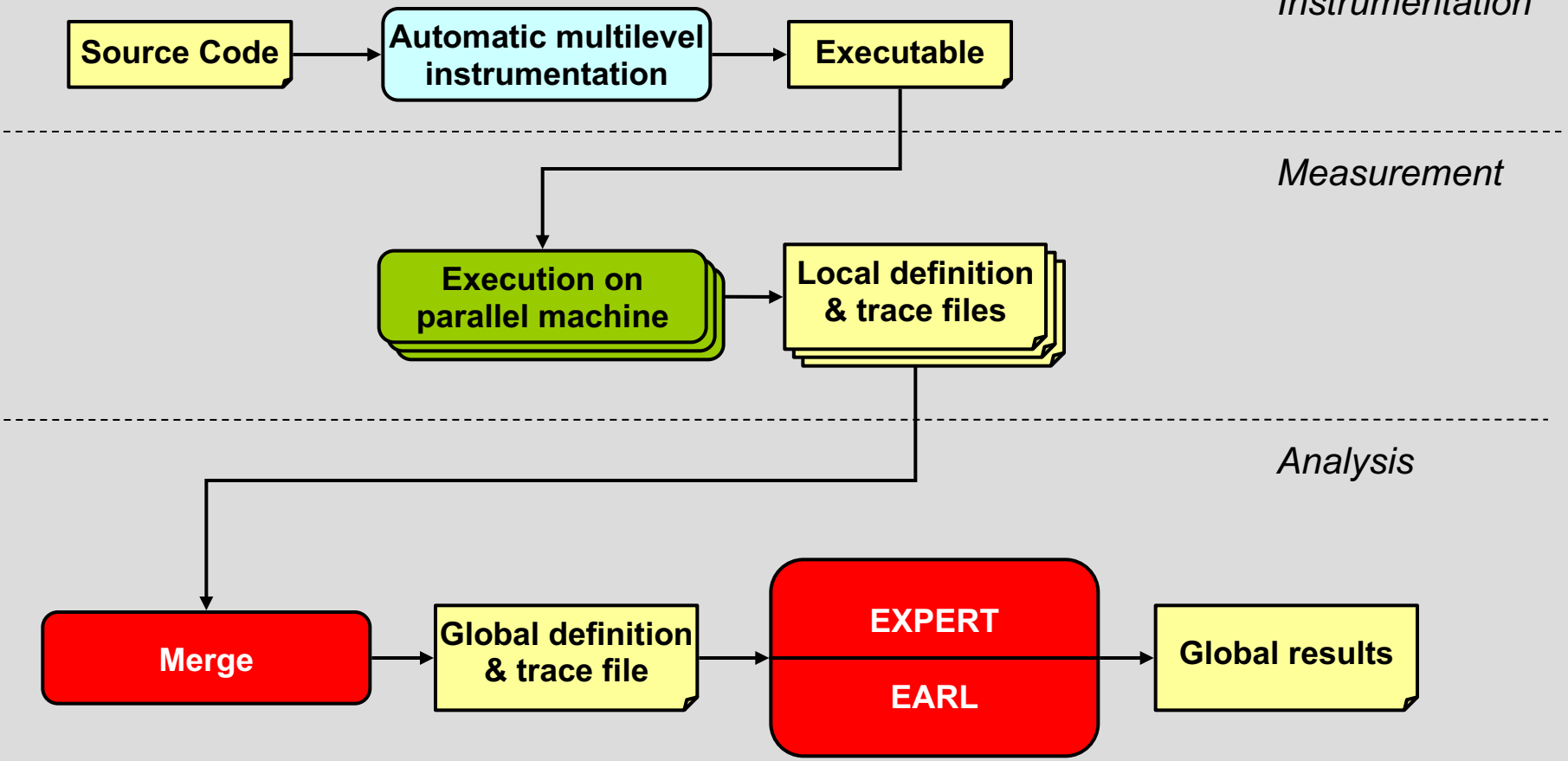
Refill from NW, NE, SE, SW

- Late sender combined with change of message direction
- Topological knowledge needed to recognize direction change
- Caveat: message direction has two components
  - Special case: border processes



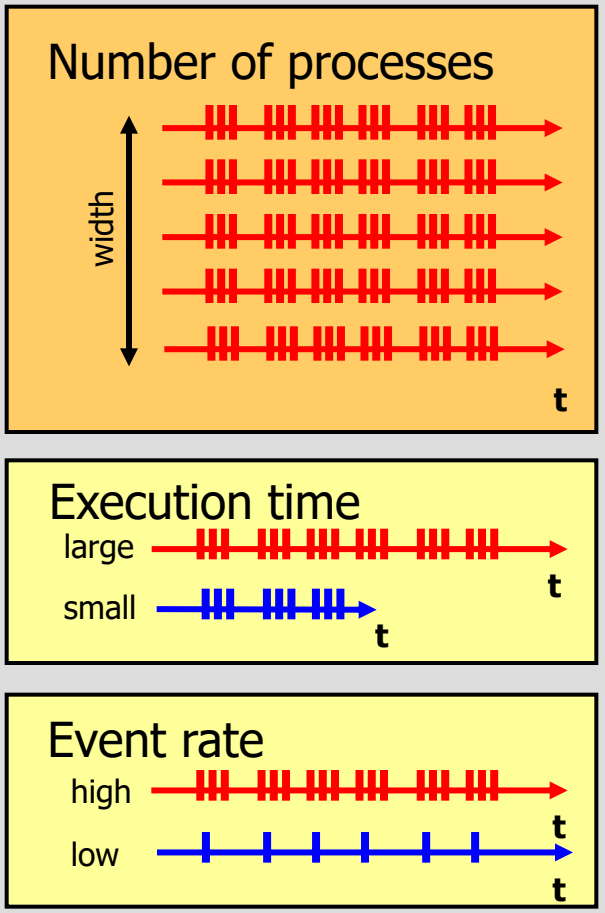



# Analysis process

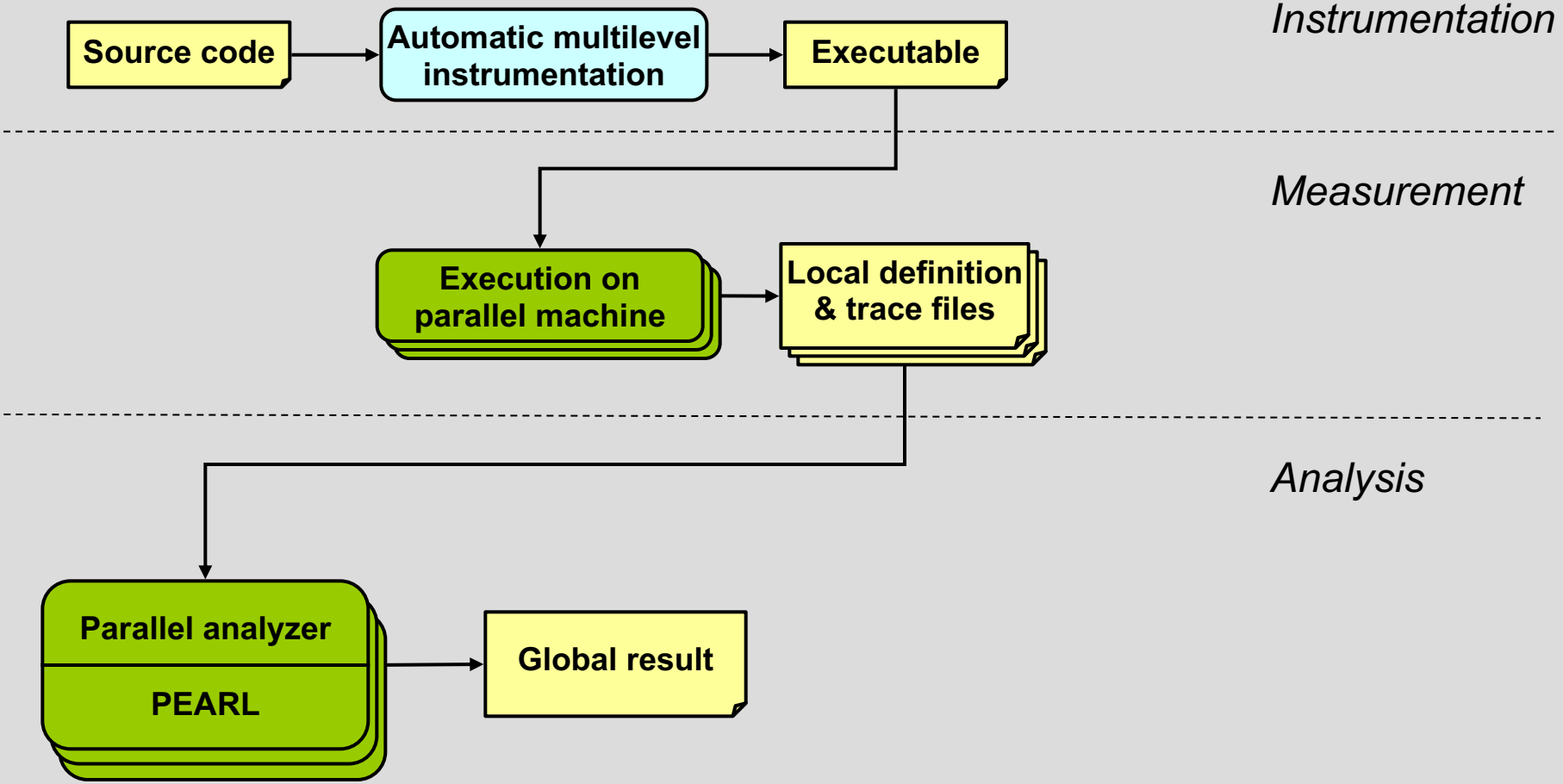


# Trace size limits scalability

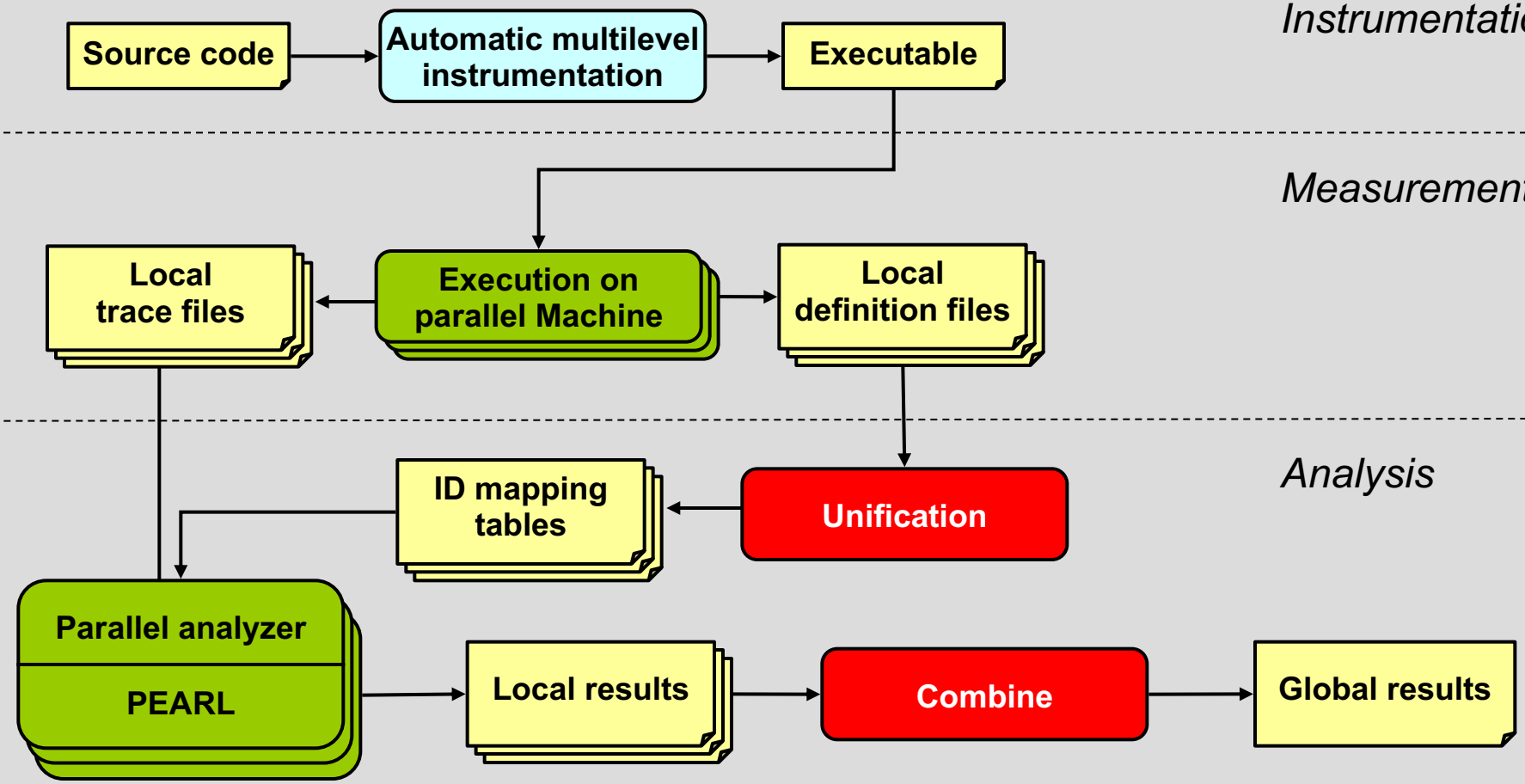
- Serially analyzing a single and potentially large global trace file does not scale to 1000s of processors
- Even if locality is exploited, main memory might be insufficient to store current working set
- Amount of trace data might not fit into single file



# Parallel analysis process



# Current prototype



scalasca

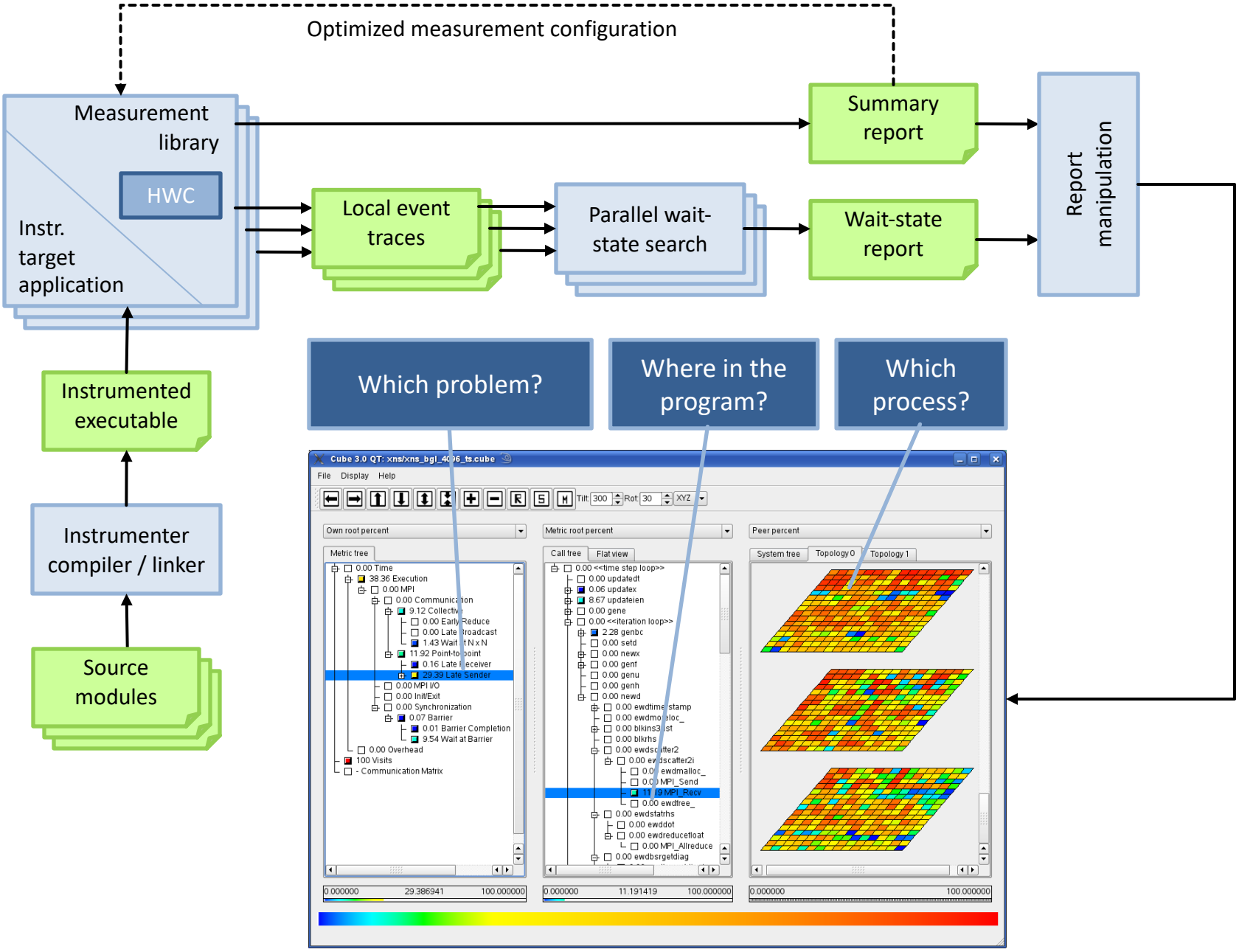
- Scalable performance-analysis toolset for parallel codes
- Integrated performance analysis process
  - Performance overview on call-path level via runtime summarization
  - In-depth study of application behavior via event tracing
  - Switching between both options without recompilation or relinking
- Supported programming models
  - MPI-1, MPI-2 one-sided communication
  - OpenMP (basic features)
- Available under the New BSD open-source license
  - http://www.scalasca.org/

Joint project of

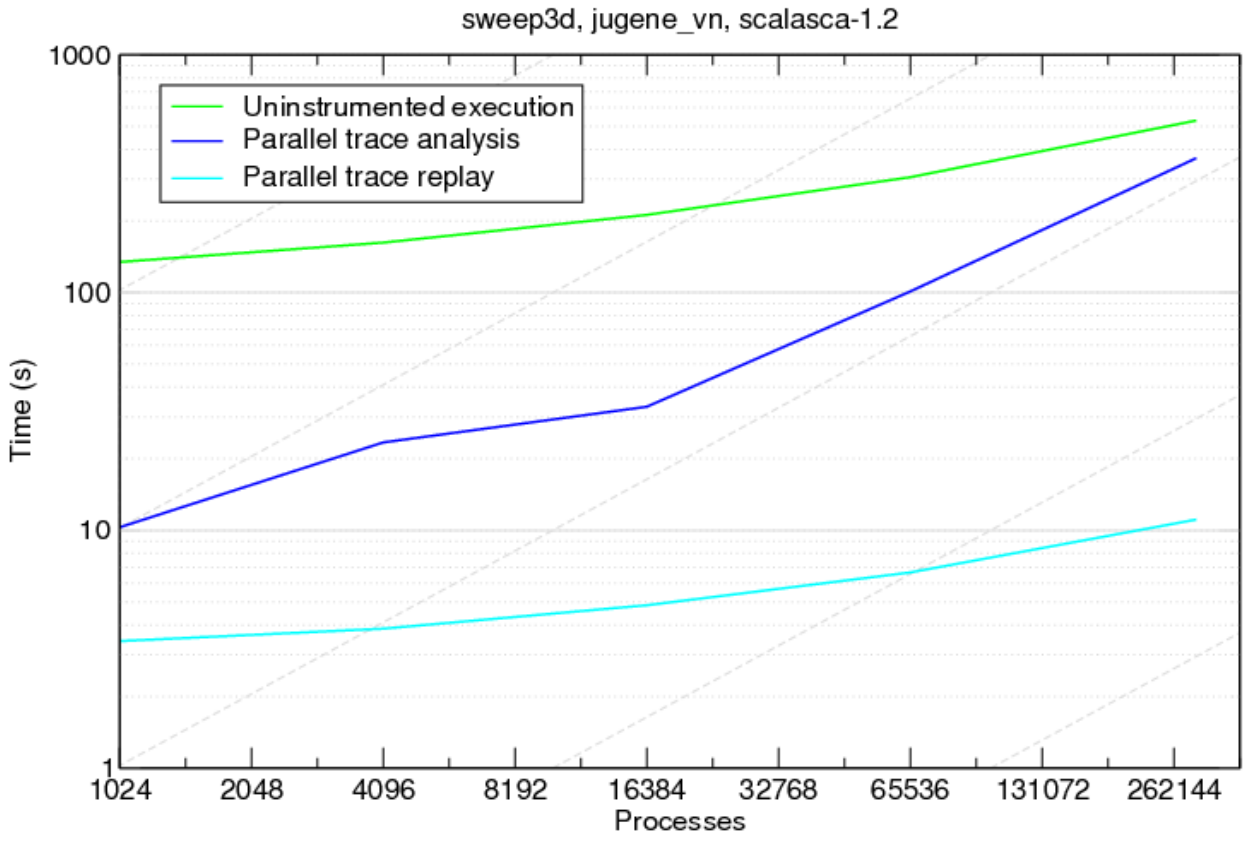
JÜLICH
FORSCHUNGSZENTRUM

German Research School
for Simulation Sciences

# The team (2010)

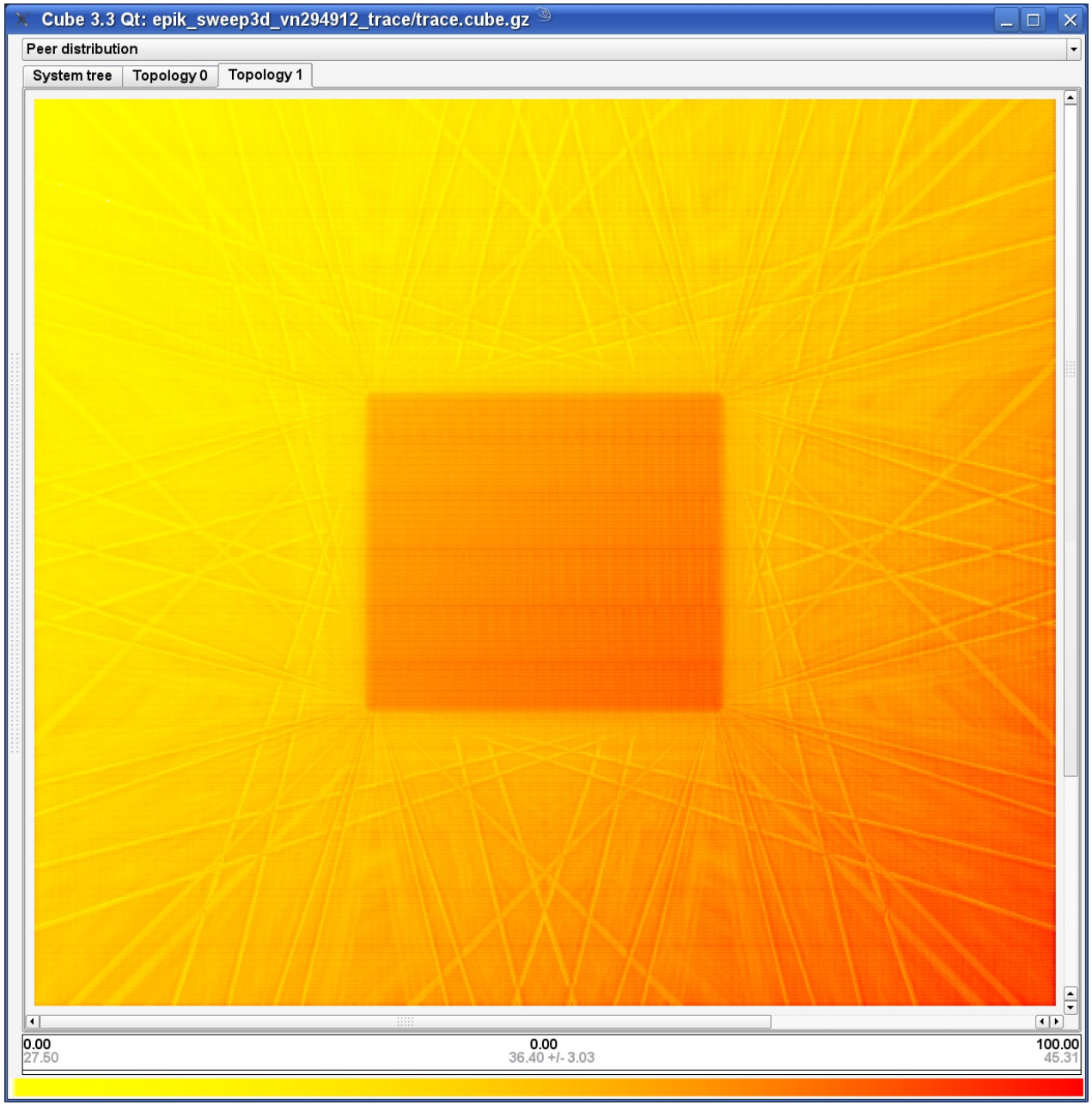

Optimized measurement configuration
Measurement library
HWC
Instr. target application
Local event traces
Parallel wait-state search
Summary report
Wait-state report
Report manipulation
Instrumented executable
Instrumenter compiler / linker
Source modules
Which problem?
Where in the program?
Which process?
Cube 3.0 QT: xns/xns_bgl_4096_ts.cube
File Display Help
Own root percent
Metric root percent
Peer percent
Metric tree
Call tree
Flat view
System tree
Topology 0
Topology 1
0.00 Time
38.36 Execution
0.00 MPI
0.00 Overhead
100 Visits
- Communication Matrix
0.00 MPI I/O
0.00 Init/Exit
0.00 Synchronization
0.07 Barrier
0.01 Barrier Completion
9.54 Wait at Barrier
0.00 <<time step loop>>
0.00 updatedt
0.06 updatex
8.67 updateien
0.00 gene
0.00 <<iteration loop>>
2.28 genbc
0.00 setd
0.00 newx
0.00 genf
0.00 genu
0.00 genh
0.00 newd
0.00 ewdstatrhs
0.00 ewddot
0.00 ewdreducefloat
0.00 MPI_Allreduce
0.00 ewdbsrgetdiag
0.00 ewdfree_
MPI_Recv
MPI_Send
0.000000
29.386941
100.000000
11.191419

# Scalability of parallel wait-state search (SWEEP3D)

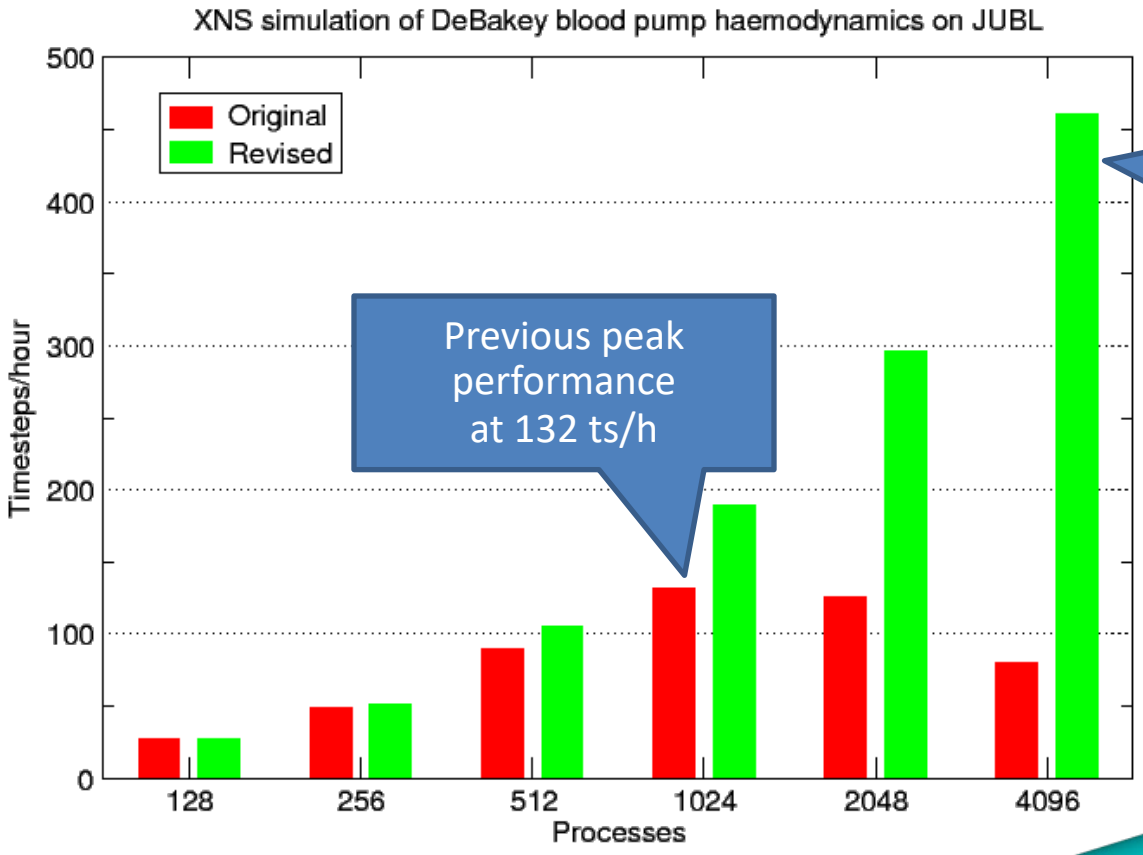


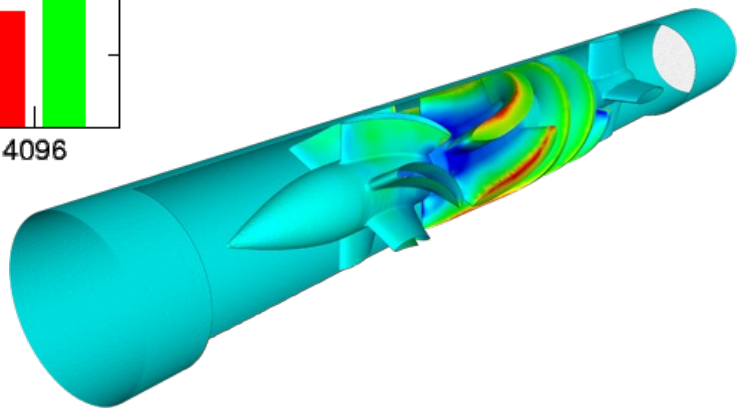
# Sweep3D
# Late sender

- Grid of 576 x 512 processes



# Redundant messages in XNS CFD code



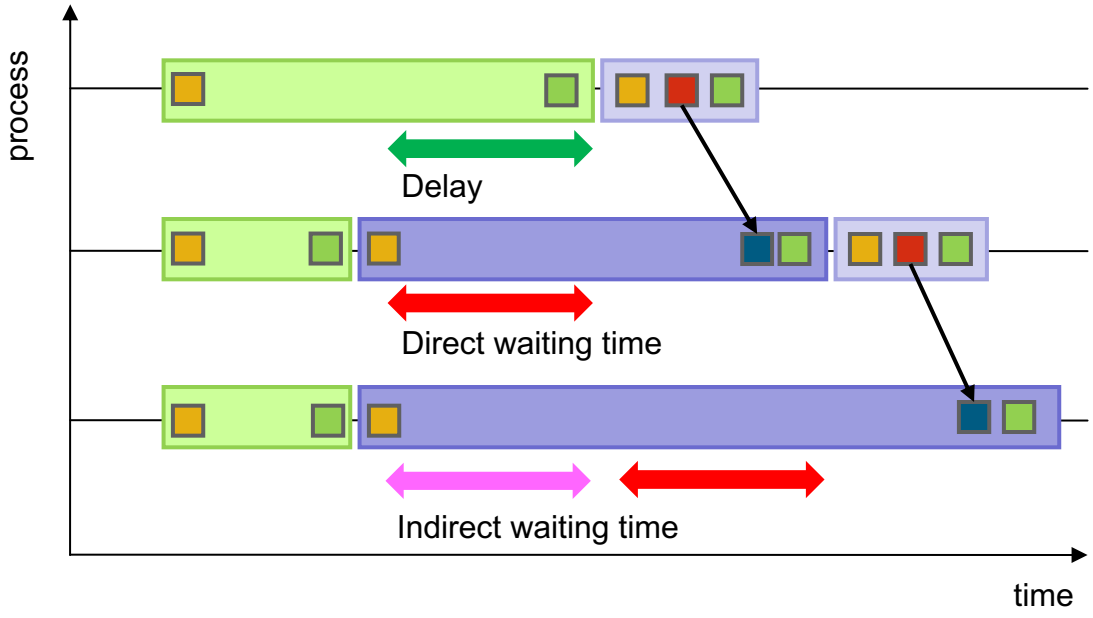



RWTH AACHEN UNIVERSITY

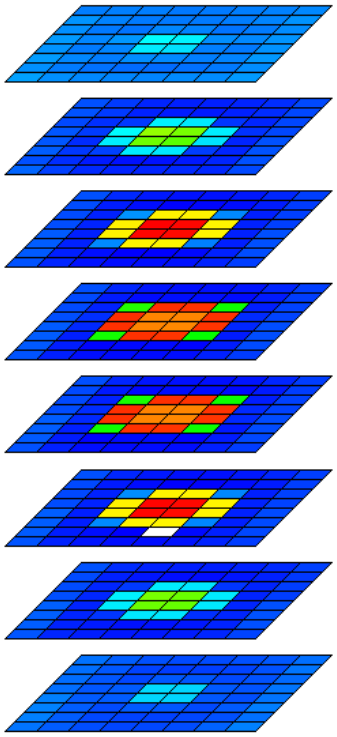
# Delay analysis

[Böhme et al., ICPP 2010]

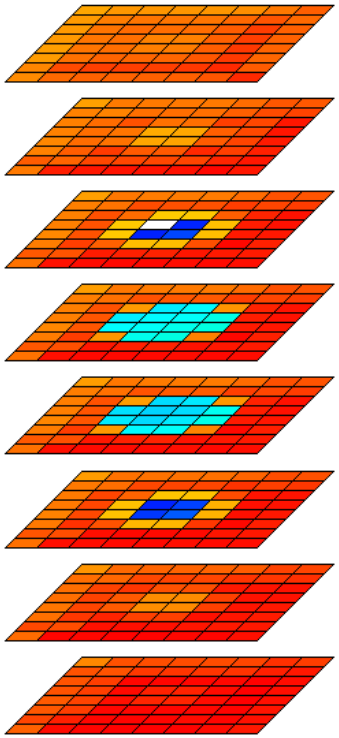
- Delay counterpart of waiting time
- Distinction between direct and indirect waiting times
- Essentially scalable version of  Meira Jr. et al.
- Analysis attributes costs of wait states to delay intervals
  - Requires backward replay



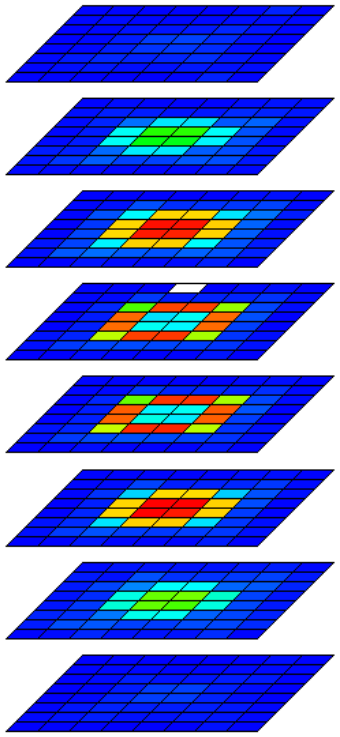
# Origin of delay costs in Zeus-MP/2



Computation



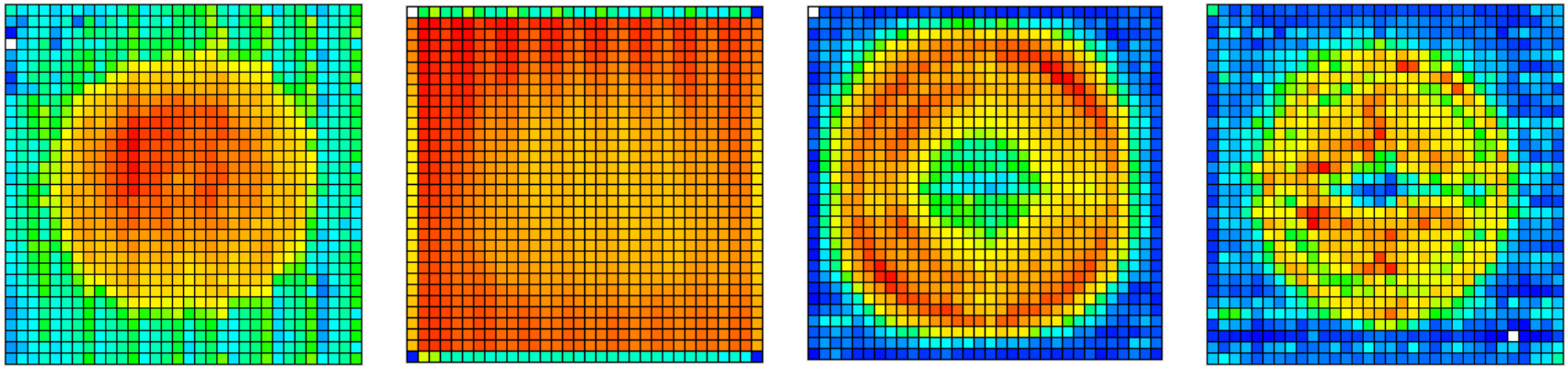
Waiting time



Delay costs

# Delay analysis of code Illumination

- Particle physics code (laser-plasma interaction)
- Delay analysis identified inefficient communication behavior as cause of wait states



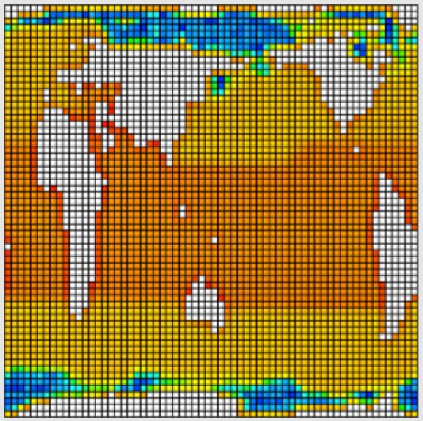
Computation

Propagating wait states:
Original vs. optimized code

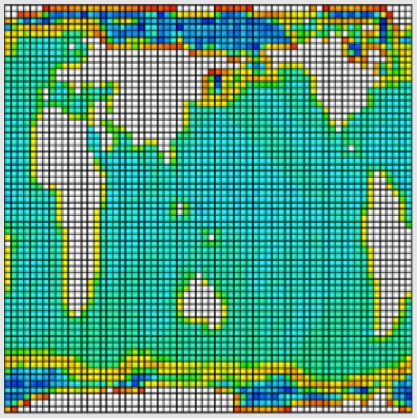
Costs of direct delay
in optimized code

German Research School
for Simulation Sciences

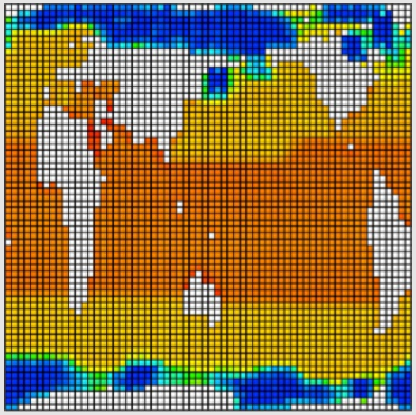
# Wait-state analysis of CICE



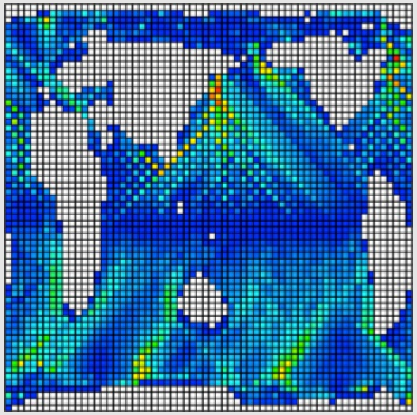
Late sender



Direct waiting time



Indirect waiting time



Delay costs

Method: bi-directional replay of event traces

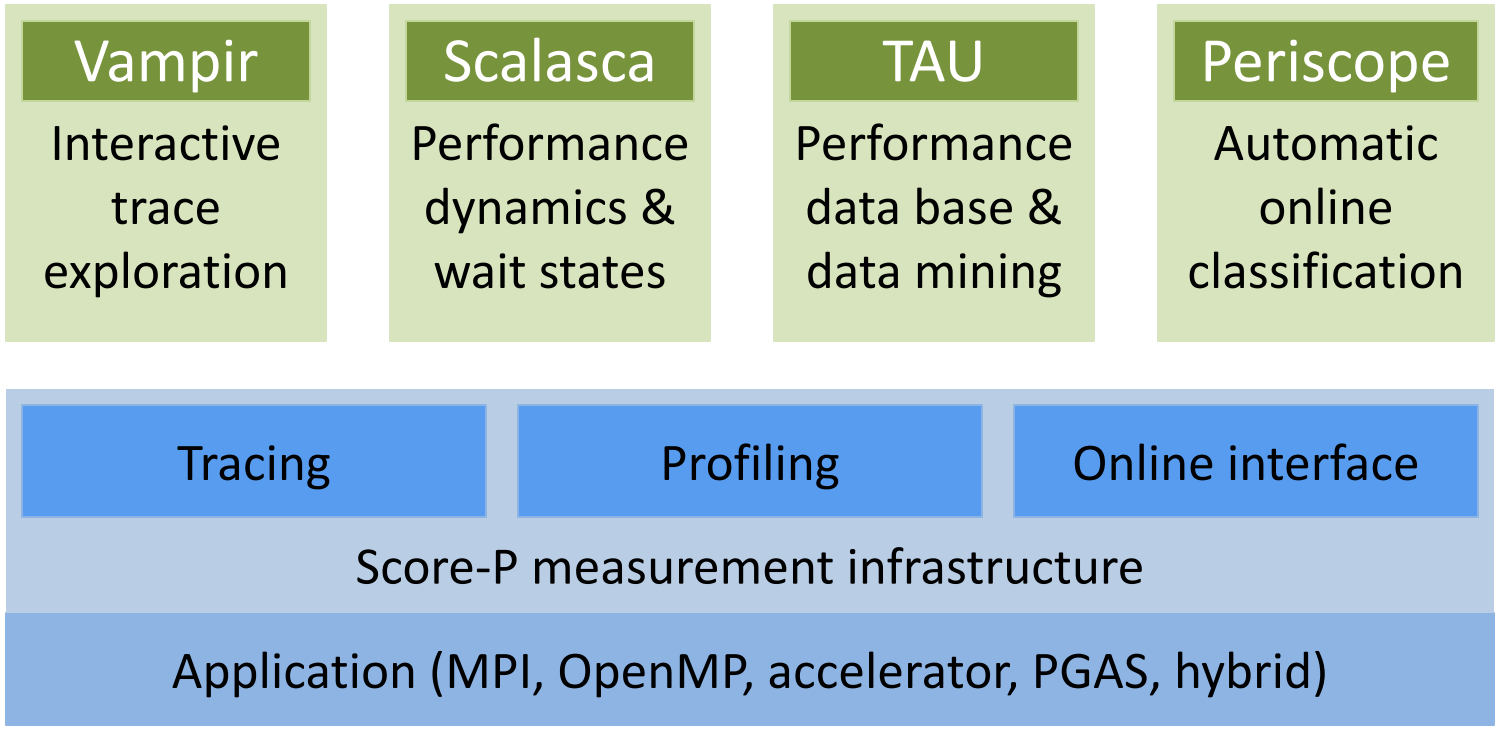
Illustrated propagation of wait states as a result of suboptimal load balancing

Motivated load-balancing simulator

## The virtual institute in a…

- Partnership to develop advanced programming tools for complex simulation codes
- Goals
  - Improve code quality
  - Speed up development
- Activities
  - Tool development and integration
  - **Training**
  - Support
  - Academic workshops
- www.vi-hps.org

# Score-P measurement system




JÜLICH FORSCHUNGSZENTRUM

German Research School for Simulation Sciences

gns

Technische Universität München

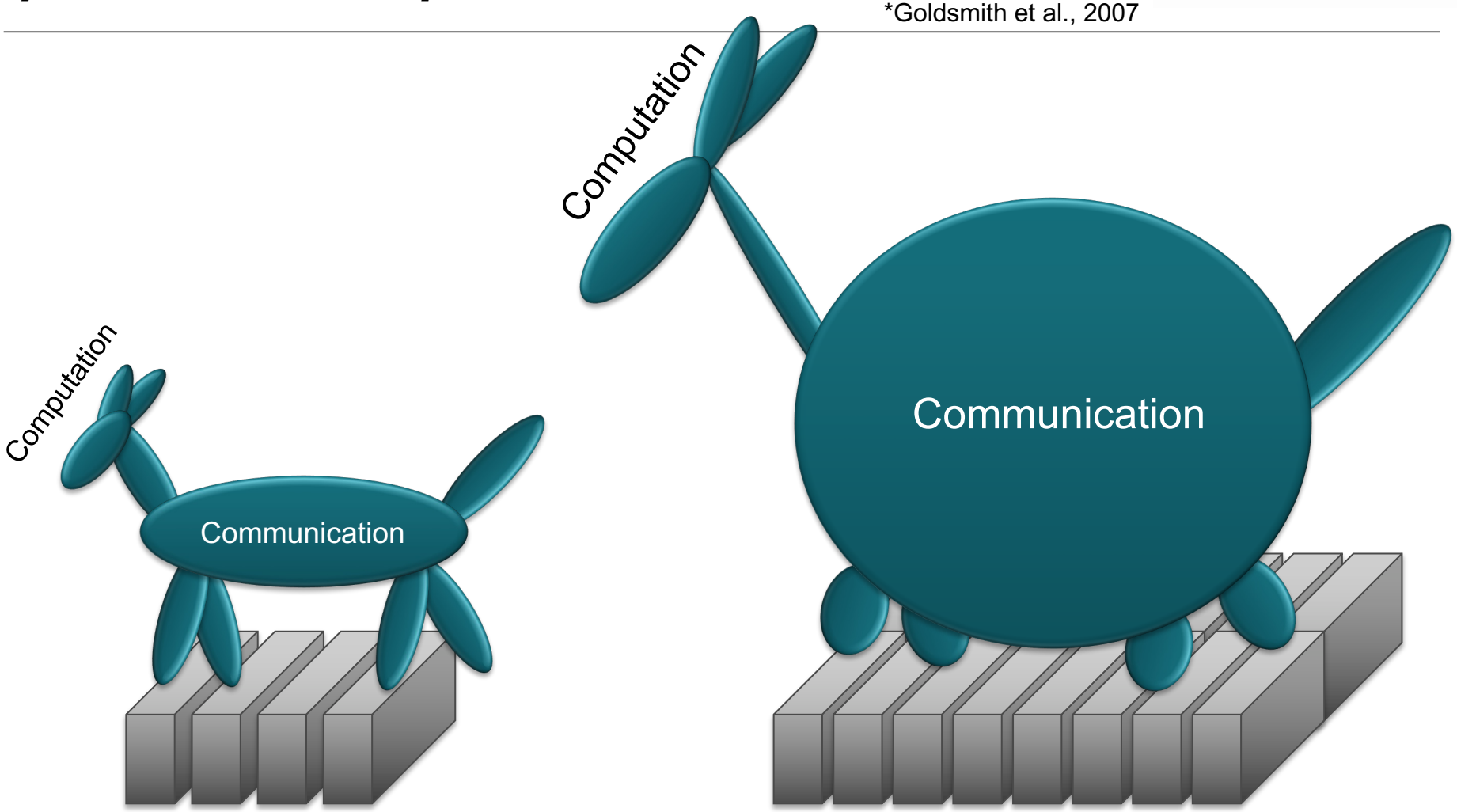
RWTH AACHEN UNIVERSITY

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITY OF OREGON

# Scaling your code can harbor *performance surprises**…

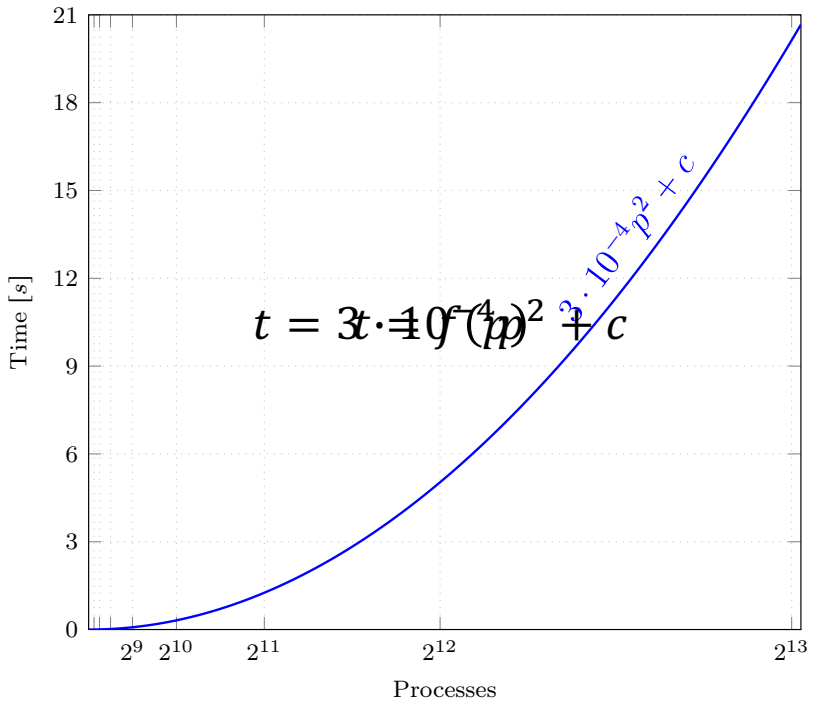
*Goldsmith et al., 2007



# Performance model

TECHNISCHE UNIVERSITÄT DARMSTADT

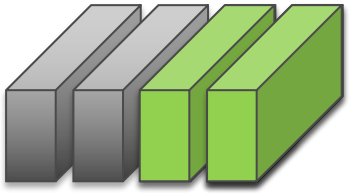
Formula that expresses a relevant performance metric as a function of one or more execution parameters



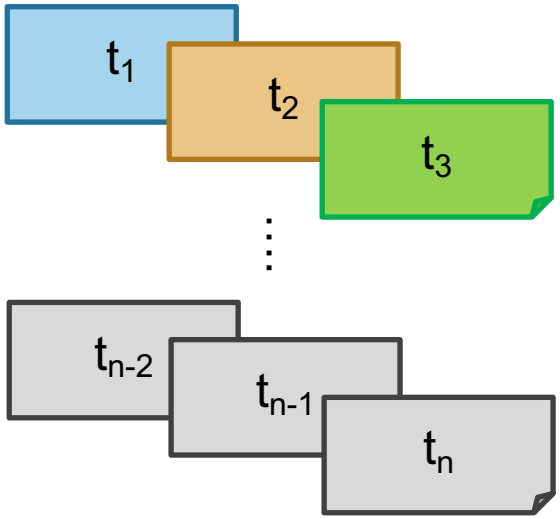

Analytical (i.e., manual) creation challenging for entire programs

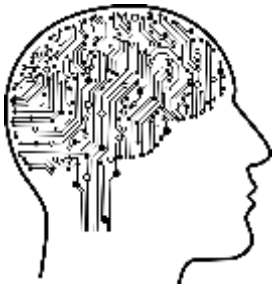
- Identify kernels
  - Incomplete coverage
- Create models
  - Laborious, difficult

# Empirical performance modeling



Performance measurements with different execution parameters $\mathbf{x_1,\ldots,x_n}$



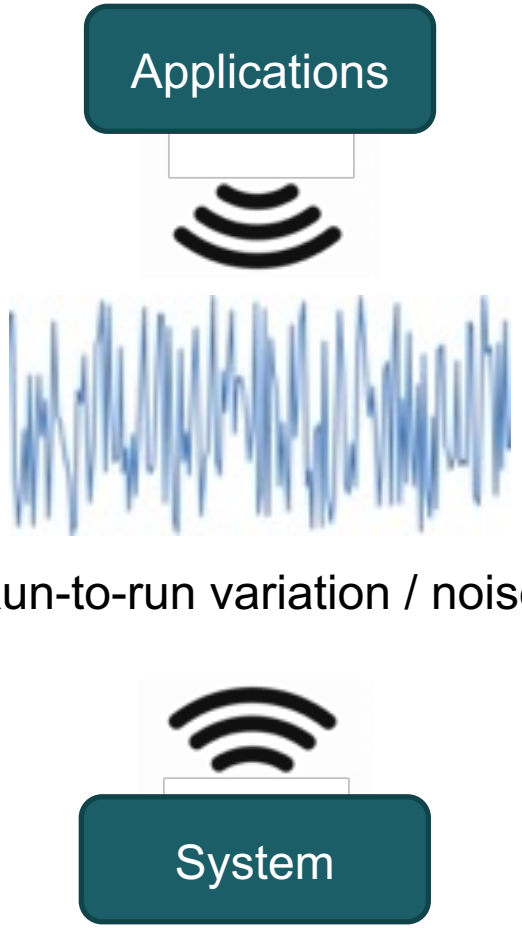



Machine learning

$$t = f(x_1, \ldots, x_n)$$

Alternative metrics: FLOPs, data volume…

# Challenges




Run-to-run variation / noise

Cost of the required experiments

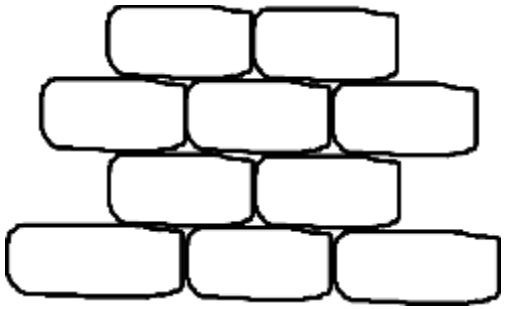
# How to deal with noisy data

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduce **prior** into learning process
  - Assumption about the probability distribution generating the data

Time

~



Effort

- Computation
- Memory access
- Communication
- I/O

# Performance model normal form (PMNF)

$$f(x) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot log_2^{j_k}(x)$$

Single parameter
[Calotoiu et al., SC13]

Parameter selection → Search space configuration → Linear regression + cross-validation → Quality assurance

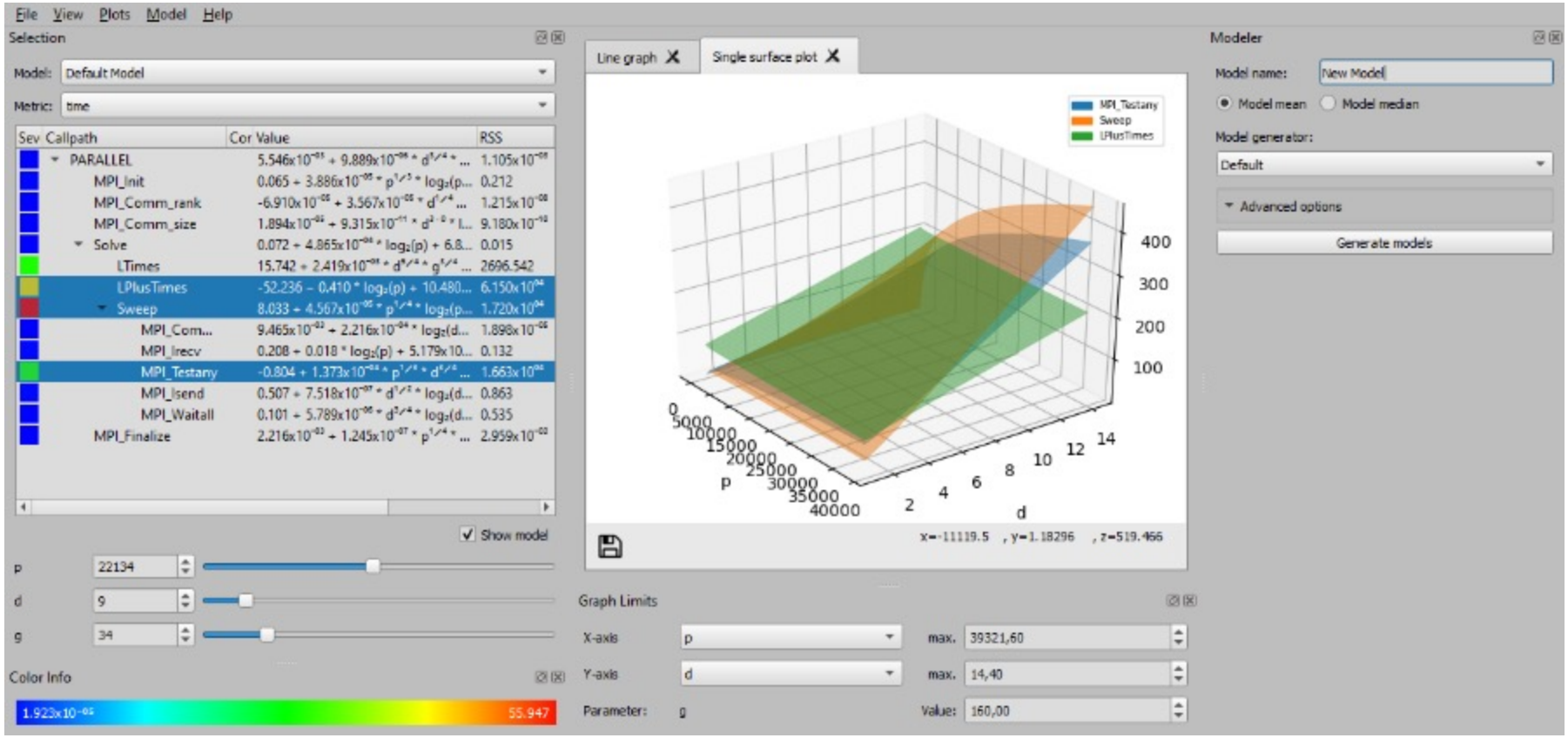
$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_l^{i_{kl}} \cdot log_2^{j_{kl}}(x_l)$$

Multiple parameters [Calotoiu et al., Cluster'16]

Heuristics to reduce search space

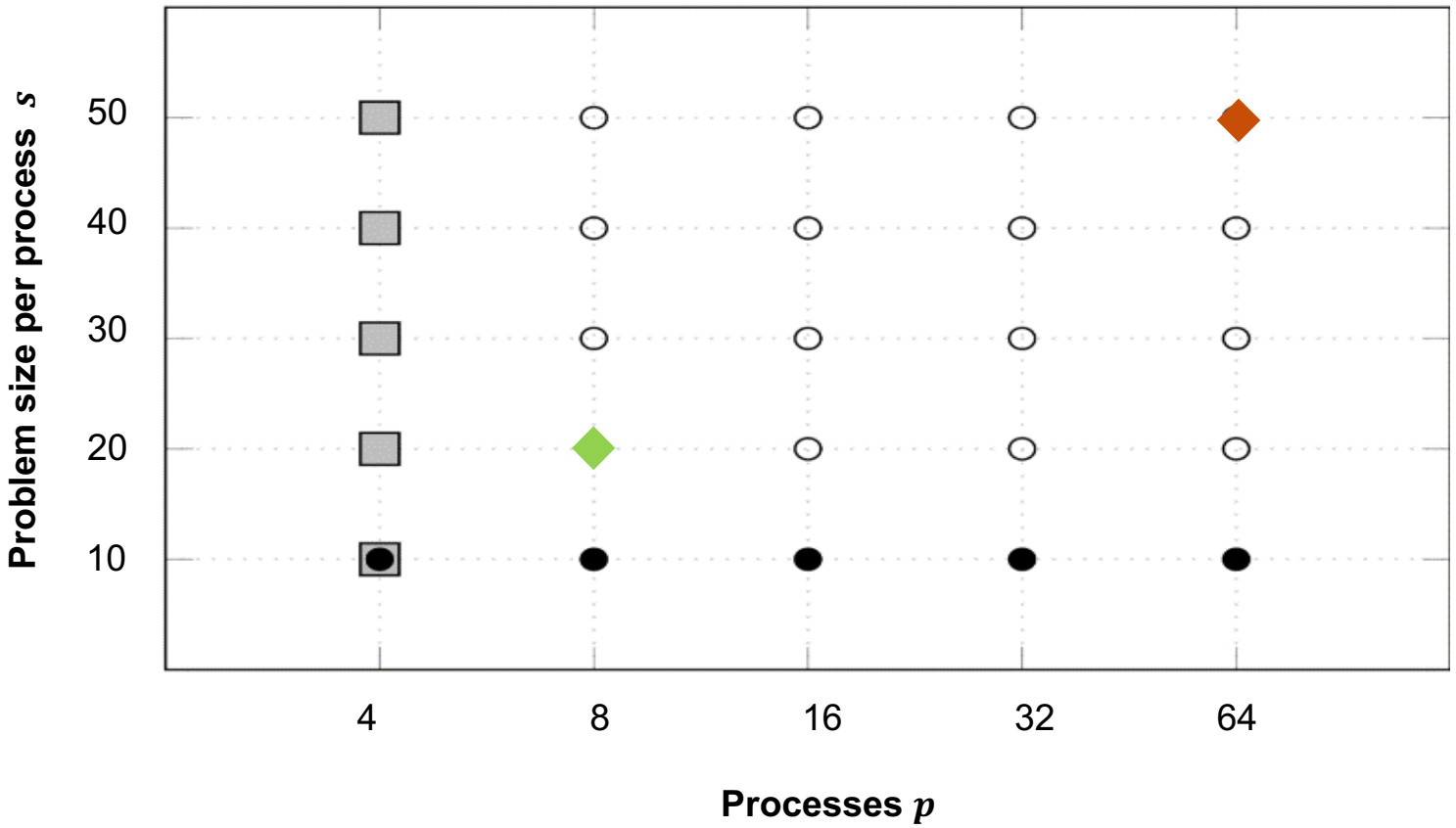
# Extra-P 4.0

TECHNISCHE UNIVERSITÄT DARMSTADT




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

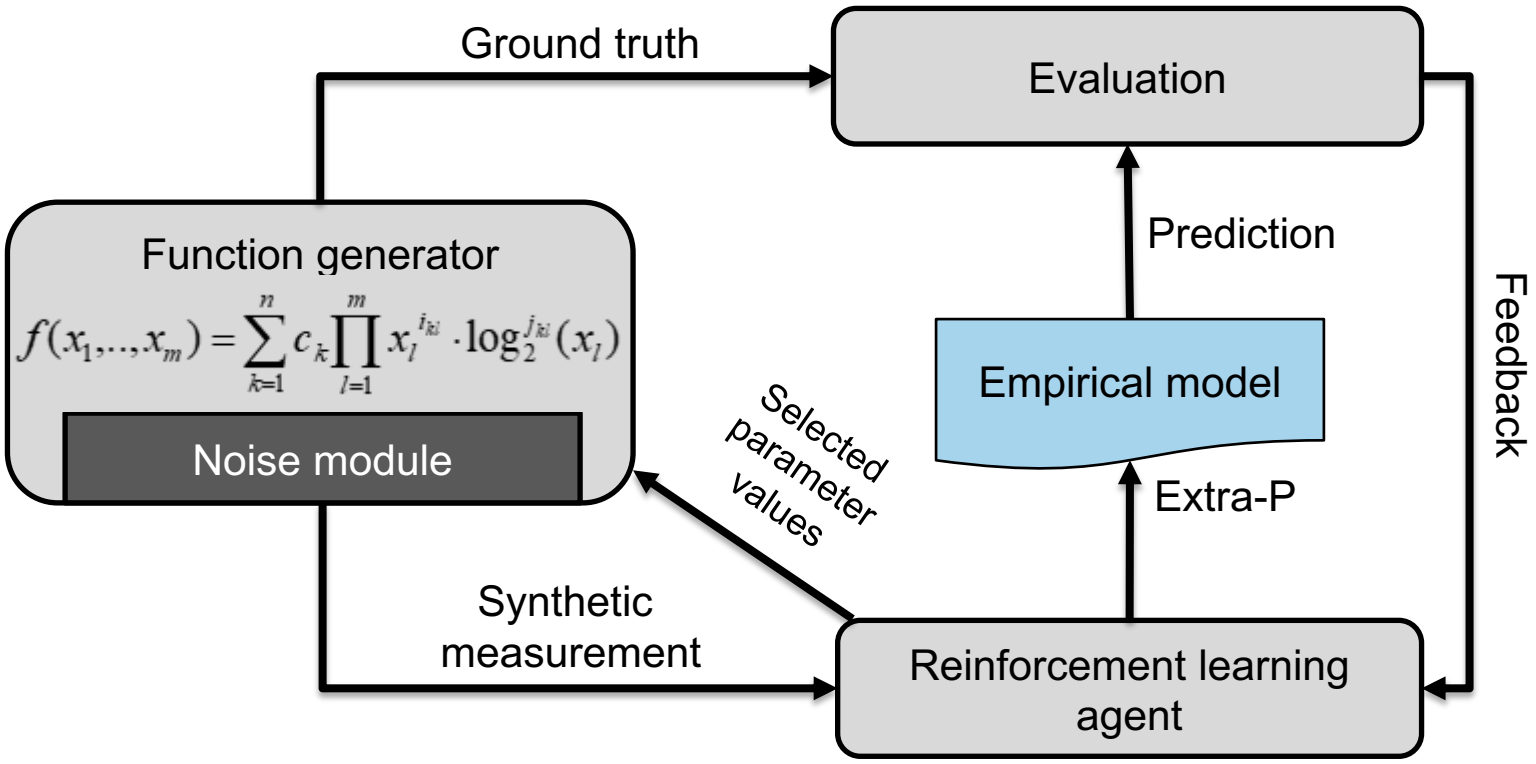
# How many data points do we really need?



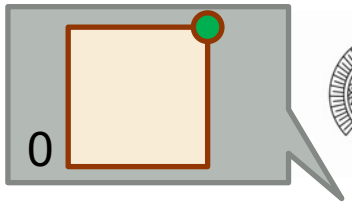
# Learning cost-effective sampling strategies

[Ritter et al., IPDPS'20]

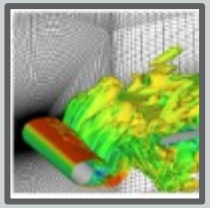
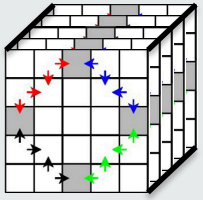
TECHNISCHE UNIVERSITÄT DARMSTADT



# Case studies



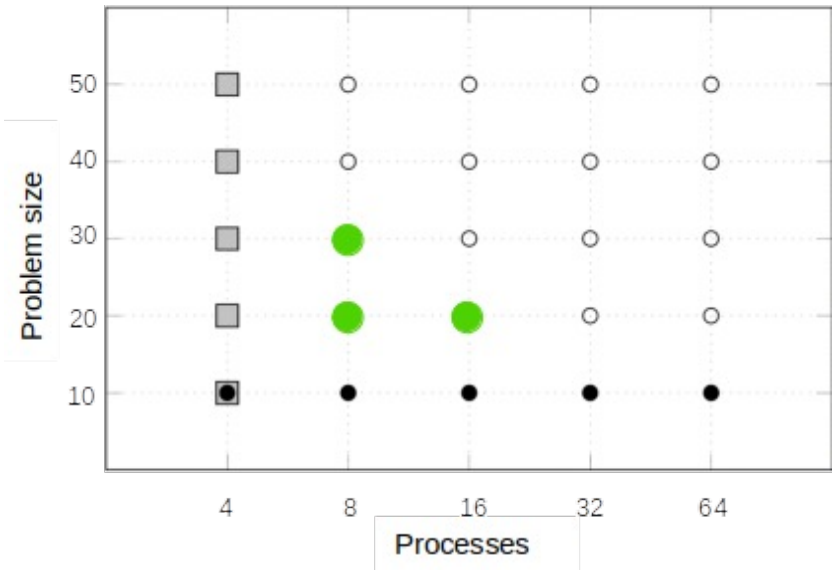

TECHNISCHE UNIVERSITÄT DARMSTADT

| Application | #Parameters | Extra points | Cost savings [%] | Prediction error [%] |
|---|---|---|---|---|
| FASTEST  | 2 | 0 | 70 | 2 |
| Kripke  | 3 | 3 | 99 | 39 |
| Relearn  | 2 | 0 | 85 | 11 |

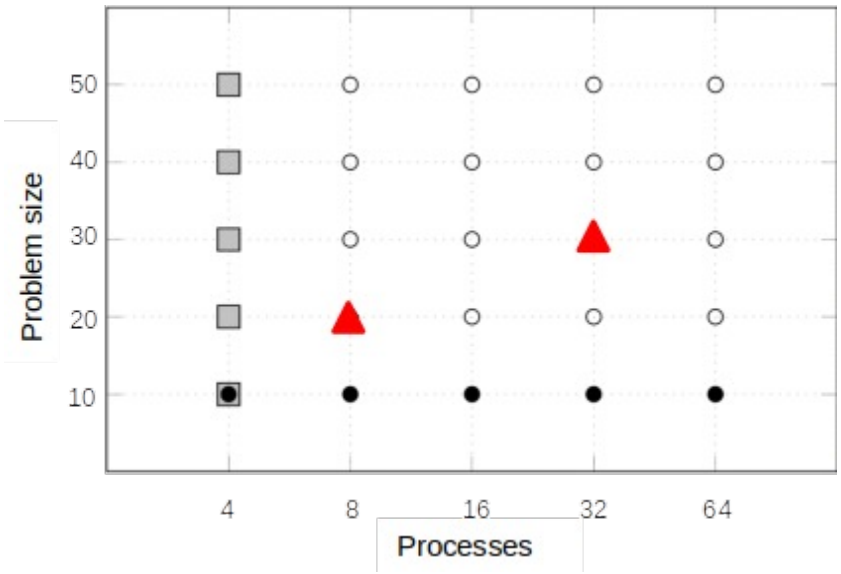
# Optimized measurement point selection

TECHNISCHE UNIVERSITÄT DARMSTADT

Sparse




Better?

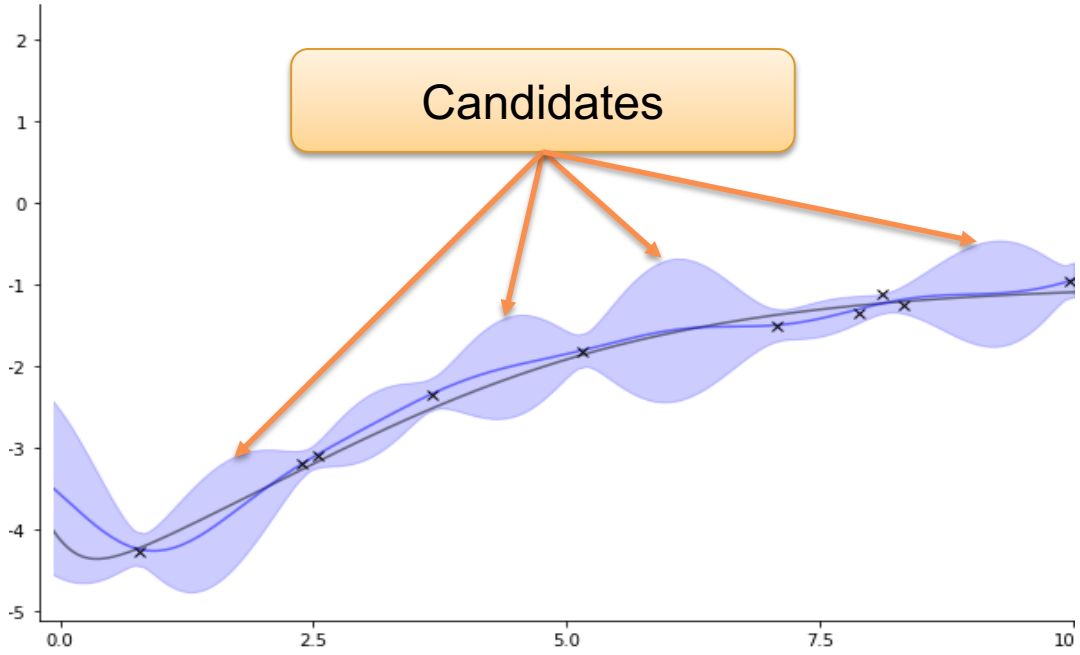


# Optimized measurement point selection
## via Gaussian Process Regression (GPR)

TECHNISCHE UNIVERSITÄT DARMSTADT

Idea: Use covariance function

High variance → High uncertainty → Promising measurement




Measurement costs have to be considered!

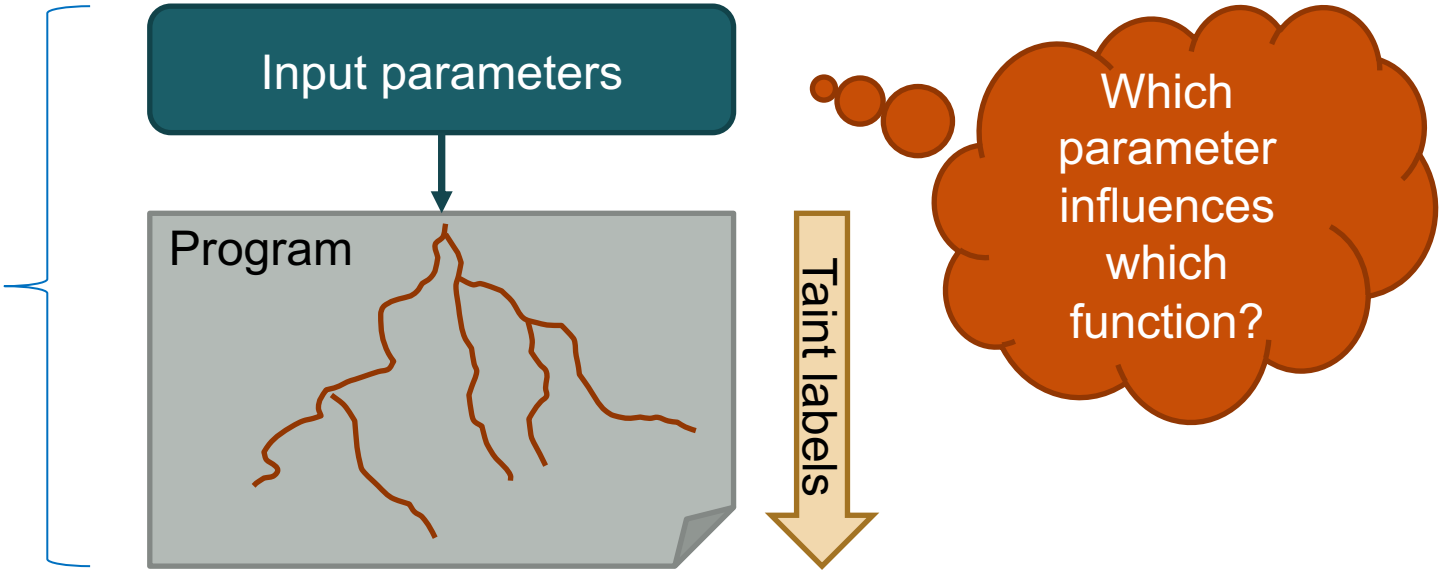
Cost-benefit calculation

# Parameter selection
[Copik et al, PPoPP'21]

TECHNISCHE UNIVERSITÄT DARMSTADT

- The more parameters the more experiments
- Modeling parameters without performance impact is harmful

**Taint analysis**



# Case study – LULESH & MILC
## Influence of program parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

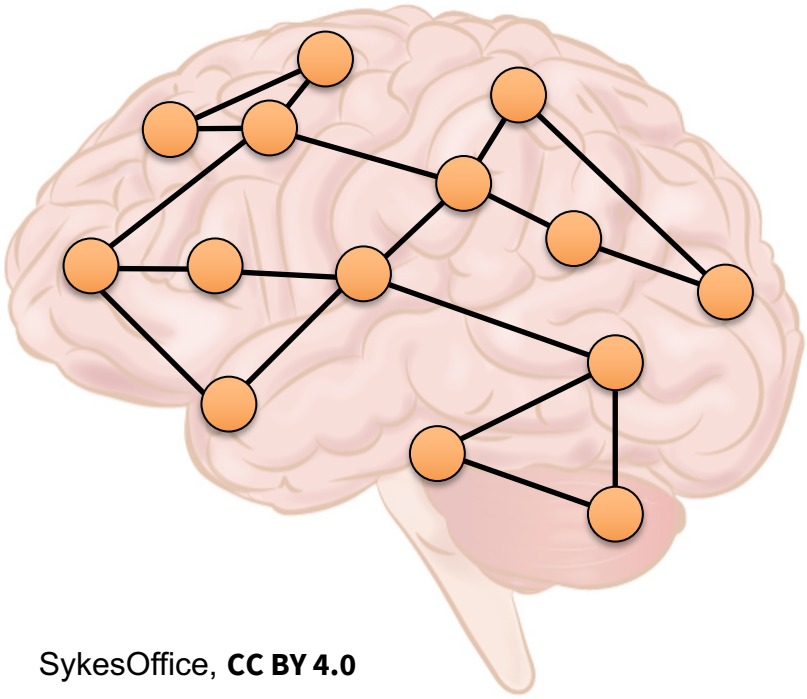
| LULESH | Total | p | size | regions | iters | balance | cost | p, size |
|---|---|---|---|---|---|---|---|---|
| Functions | 349 | 2 | 40 | 15 | 1 | 1 | 2 | 40 |
| Loops | 275 | 2 | 78 | 29 | 1 | 1 | 2 | 78 |

| MILC | Total | p | size | trajecs | warms steps | nrest. niter | mass, beta nfl. | u0 | p, size |
|---|---|---|---|---|---|---|---|---|---|
| Functions | 621 | 54 | 53 | 12 | 9 | 6 | 1 | 4 | 56 |
| Loops | 874 | 187 | 161 | 39 | 31 | 15 | 1 | 7 | 196 |

# Relearn

[Rinke et. al, JPDC'18]

TECHNISCHE UNIVERSITÄT DARMSTADT

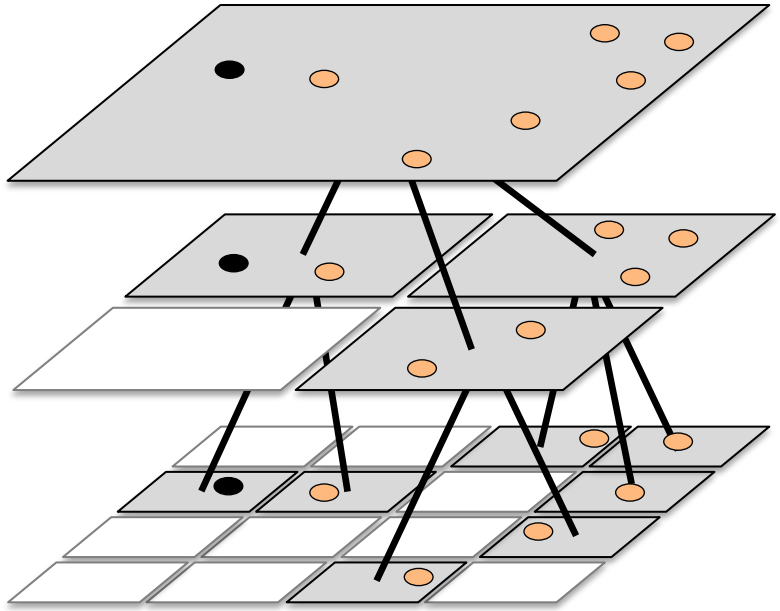


SykesOffice, **CC BY 4.0**

- Scalable algorithm to simulate structural plasticity in the brain
- Adaptation of Barnes-Hut algorithm (astrophysics)

# Complexity consideration

- O(n * log2n) = O(n * log n * log n):
  - Barnes–Hut at every level:
    O(n * log n)
  - Tree depth: O(log n)
- Parallel complexity:
  $O(n/p * \log^2 n + p)$



# Media coverage

TECHNISCHE UNIVERSITÄT DARMSTADT

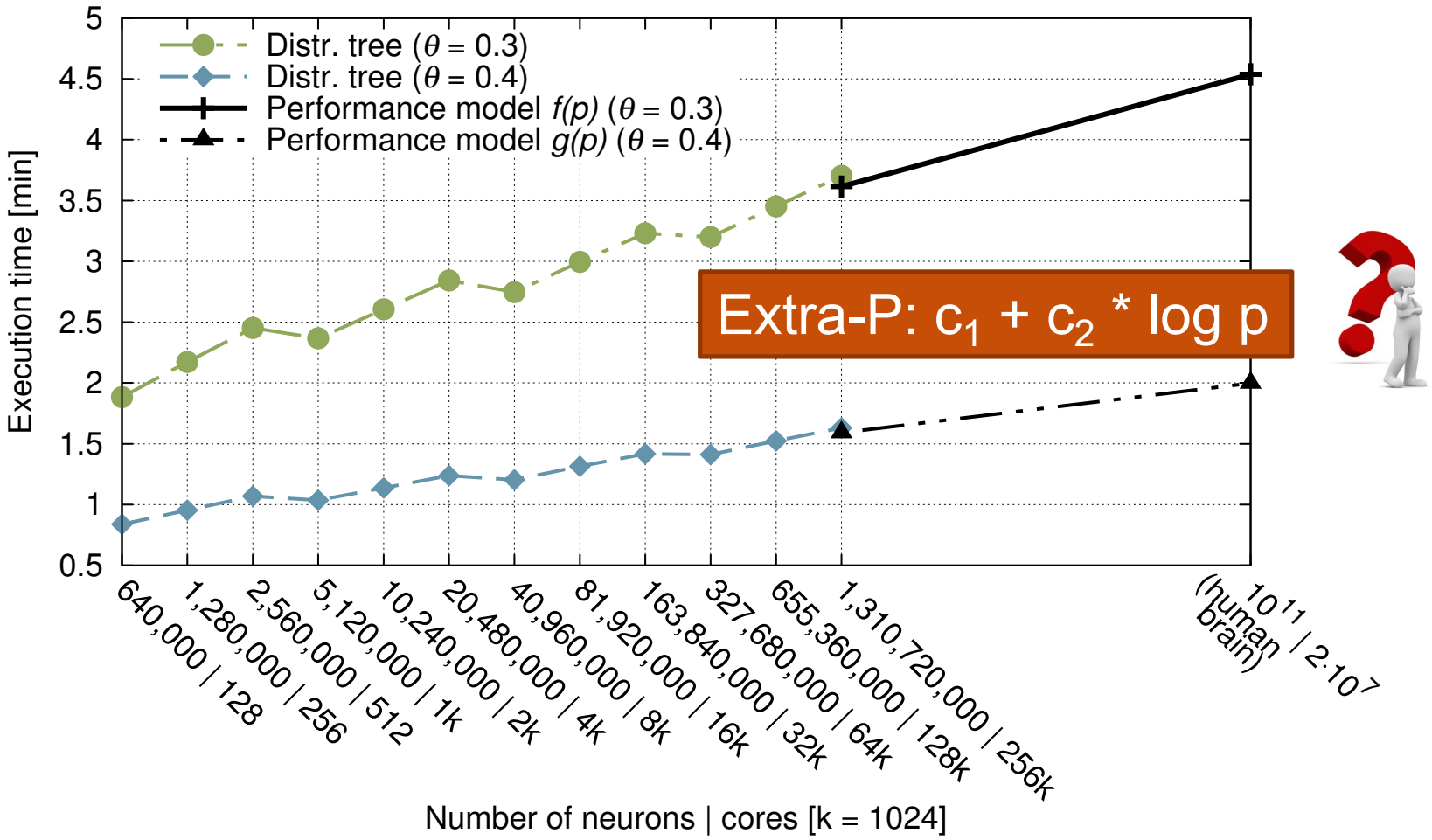
## TV - SAT.1 Regionalmagazin für Rheinland-Pfalz und Hessen

https://www.1730live.de/wissenschaftler-wollen-gehirn-nachbauen/

SCIENCENODE™

#6 of 2019

# Execution time
## One connectivity update



# Sharp complexity for scalable algorithm

[Czappa et al, JPDC'23]

TECHNISCHE UNIVERSITÄT DARMSTADT

- Depending on acceptance criterion *Θ*, subsequent Barnes–Hut steps have constant complexity
  - O(n * (log n + log n)) = O(n * log n)
- Overall complexity for practical scenarios:

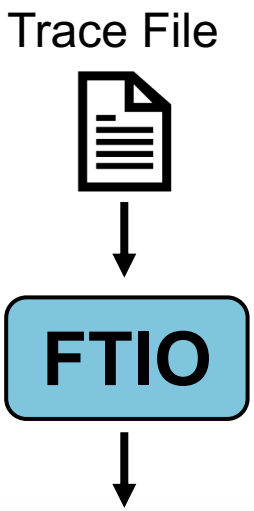
  **O(n/p * log n + p)**

# FTIO: Frequency techniques for I/O

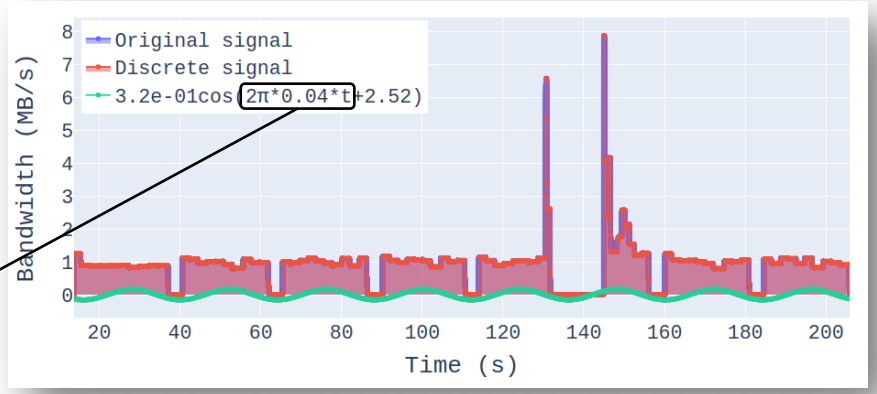
[Tarraf et al., IPDPS ’24]

TECHNISCHE UNIVERSITÄT DARMSTADT

- Captures the **period of** I/O phases
- Operates in the **frequency domain**
- Quantifies the **confidence** in the results
- Online (**prediction**) w/ low overhead and offline (**detection**)

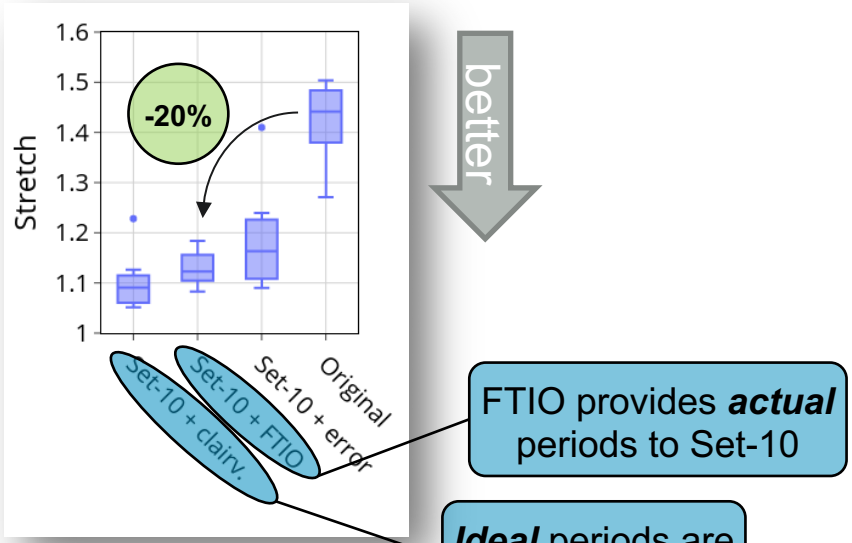



Frequency = 0.04 Hz
→ Period = 24.025 s
Confidence = 64.9% …



# Use case: I/O scheduling with IO-Sets

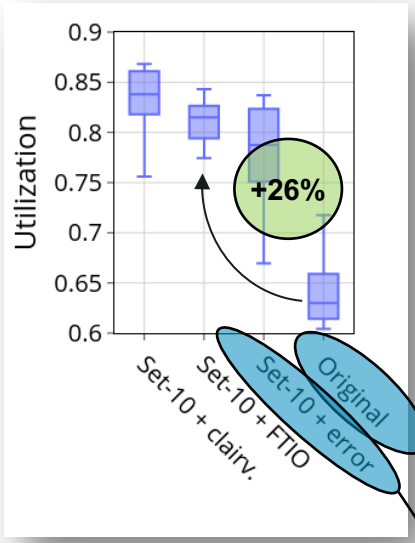
- Classify applications based on their period
  - Shared file-system access to different classes
  - Mutually exclusive access to individual jobs within the same class




FTIO provides ***actual*** periods to Set-10

***Ideal*** periods are hardcoded

Slow-down compared to running in isolation




BeeGFS without any modifications

Error (50%) is injected in the result from FTIO

Fraction of system time spent on computation

# Thank you!

TECHNISCHE UNIVERSITÄT DARMSTADT

Q&A

# Literature

TECHNISCHE UNIVERSITÄT DARMSTADT

| Tool | Paper |
|---|---|
| KOJAK/ EXPERT | • Michael Gerndt, Bernd Mohr, Felix Wolf, Mario Pantano: Performance Analysis on Cray T3E. In *Proc. of the 7th Euromicro Workshop on Parallel and Distributed Processing (PDP), Funchal, Madeira, Portugal*, pages 241–248, IEEE, February 1999.<br>• Felix Wolf, Bernd Mohr: Automatic performance analysis of hybrid MPI/OpenMP applications. Journal of Systems Architecture, 49(10-11):421–439, November 2003. |
| Scalasca | • Markus Geimer, Felix Wolf, Brian J. N. Wylie, Bernd Mohr: A scalable tool architecture for diagnosing wait states in massively parallel applications. Parallel Computing, 35(7):375–388, July 2009.<br>• Markus Geimer, Felix Wolf, Brian J. N. Wylie, Erika Ábrahám, Daniel Becker, Bernd Mohr: The Scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience, 22(6):702–719, April 2010.<br>• David Böhme, Markus Geimer, Lukas Arnold, Felix Voigtländer, Felix Wolf: Identifying the root causes of wait states in large-scale parallel applications. ACM Transactions on Parallel Computing, 3(2):Article No. 11, 24 pages, July 2016. |
| Extra-P | • Alexandru Calotoiu, Torsten Hoefler, Marius Poke, Felix Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes. In Proc. of the ACM/IEEE Conference on Supercomputing (SC13), Denver, CO, USA, pages 1–12, ACM, November 2013.<br>• Sergei Shudler, Yannick Berens, Alexandru Calotoiu, Torsten Hoefler, Alexandre Strube, Felix Wolf: Engineering Algorithms for Scalability through Continuous Validation of Performance Expectations. IEEE Transactions on Parallel and Distributed Systems, 30(8):1768–1785, August 2019. |
| FTIO | • Ahmad Tarraf, Alexis Bandet, Francieli Boito, Guillaume Pallez, Felix Wolf: Capturing Periodic I/O Using Frequency Techniques. In Proc. of the 38th IEEE International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, CA, USA, pages 1–14, IEEE, May 2024, (accepted). |

# Acknowledgment


Alexis Bandet
Nikhil Battia
Alexandru Calotoiu
Daniel Becker
Francieli Boito
David Böhme
Dominic Eschweiler
Marcin Copik
Jack Dongarra
Christian Feld
Wolfgang Frings
Markus Geimer
Alexander Geiß
Michael Gerndt
Marc-André Hermanns
Torsten Hoefler
Michael Knobloch
Daniel Lorenz
Bernd Mohr
Shirley Moore
Guillaume Pallez
Farzona Pulatova
Sebastian Rinke
Marcus Ritter
Pavel Saviankou
Martin Schulz
Christian Siebert
Sergei Shudler
Fengguan Song
Zoltán Szebenyi
Ahmad Tarraf
Brian Wiley
[…]