German Research School
for Simulation Sciences

Characterizing Imbalance in Large-Scale Parallel Programs

David Böhme

September 26, 2013

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Need for Performance Analysis Tools

- Amount of parallelism in Supercomputers keeps growing
  - Efficient resource usage depends on software performance

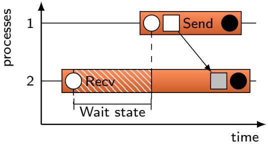
Jugene (2008): 65 536 cores

Juqueen (2013): 458 752 cores

German Research School
for Simulation Sciences

Imbalance limits parallel efficiency

Imbalance leads to wait states at synchronization points



Late-sender wait state

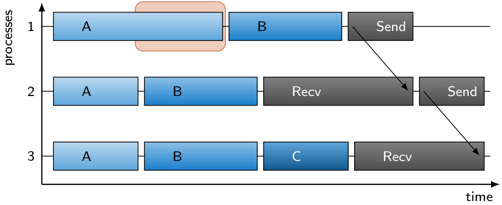
Goal: Locate inefficient parallelism and quantify its impact

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Two novel analysis methods

Root-cause analysis

Critical-path analysis



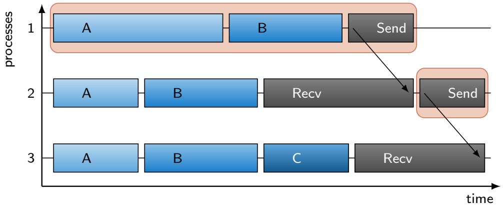
Determine delays that cause wait states.

German Research School for Simulation Sciences

Two novel analysis methods

Root-cause analysis

Critical-path analysis



Identify activities that determine program runtime.

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Outline

Parallel Performance Analysis

Root-Cause Analysis

- Concepts
- Case study

Critical-Path Analysis

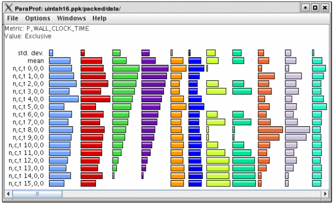
- The critical path
- Critical-path imbalance indicator
- Analysis of MPMD programs

Implementation

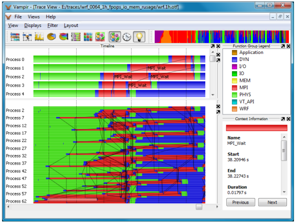
- Parallel trace replay
- Scalability evaluation

Performance analysis tools

- Tools help understand performance, but
  - Profiling provides limited insights
  - Tracing produces too much data to analyze manually



Time profile display in TAU

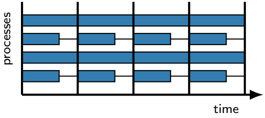


Event-trace timeline visualization in Vampir

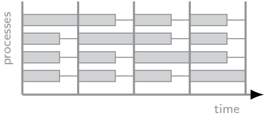
JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Imbalance analysis pitfalls

- Profilers underestimate impact of imbalance
  - Data aggregation hides dynamic imbalance effects



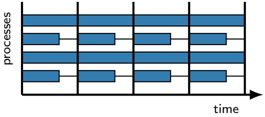
Static imbalance



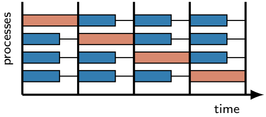
Dynamic imbalance

Imbalance analysis pitfalls

- Profilers underestimate impact of imbalance
  - Data aggregation hides dynamic imbalance effects



Static imbalance

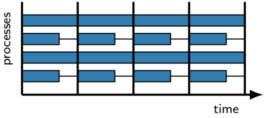


Dynamic imbalance

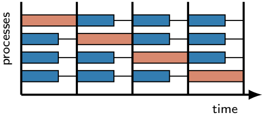
- Analysis solution needs to retain performance dynamics

Imbalance analysis pitfalls

- Profilers underestimate impact of imbalance
  - Data aggregation hides dynamic imbalance effects



Static imbalance

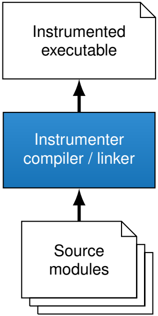


Dynamic imbalance

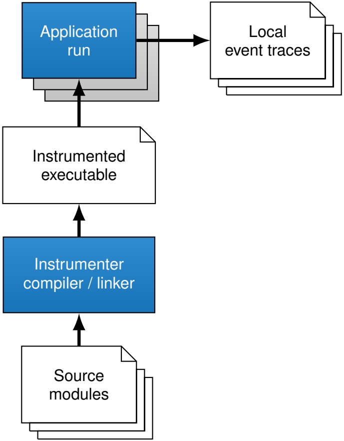
- Analysis solution needs to retain performance dynamics
- Use automatic trace analysis

scalasca

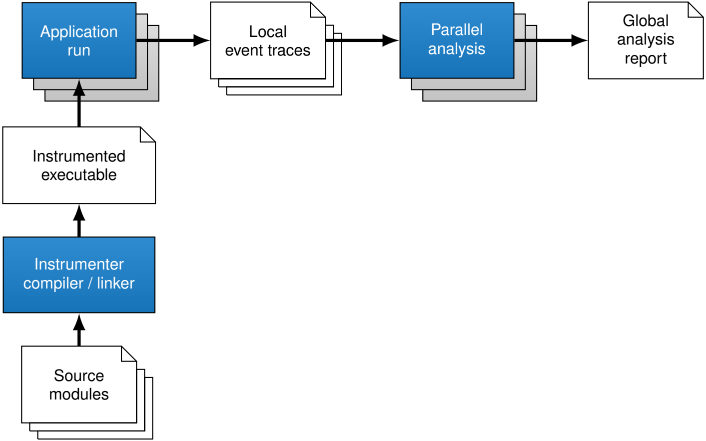
Analysis workflow



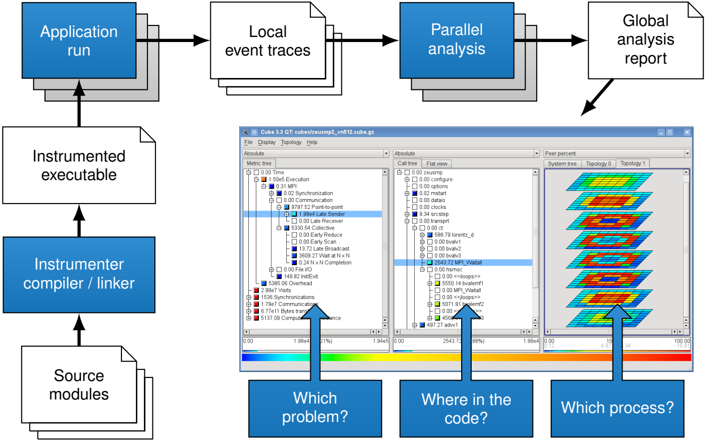
Analysis workflow



Analysis workflow



Analysis workflow



Parallel Performance Analysis

Root-Cause Analysis

- Concepts
- Case study

Critical-Path Analysis

- The critical path
- Critical-path imbalance indicator
- Analysis of MPMD programs

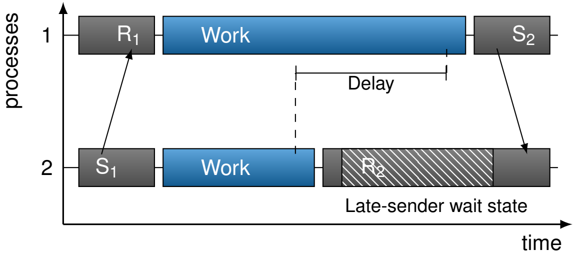
Implementation

- Parallel trace replay
- Scalability evaluation

German Research School
for Simulation Sciences

Delay as root cause of wait states

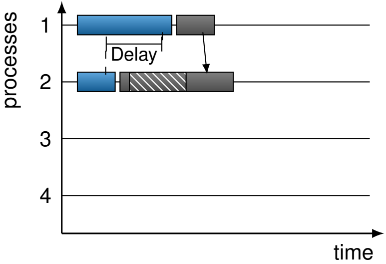
A *delay* is some additional activity on one process that causes a wait state at a synchronization point



Delay on process 1 causes a late-sender wait state on process 2

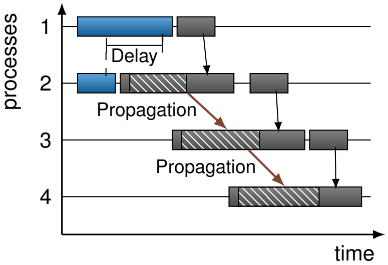
Wait-state propagation

- Wait states can propagate



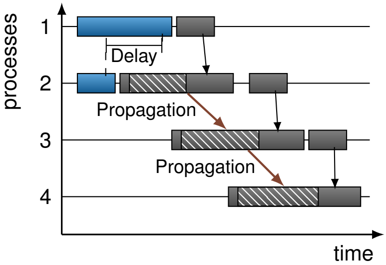
Wait-state propagation

- Wait states can propagate



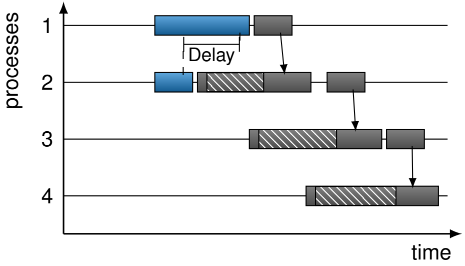
Wait-state propagation

- Wait states can propagate
- Account for propagation in analysis
  - Extended wait-state classification
  - Incorporate long-distance effects in calculation of delay costs



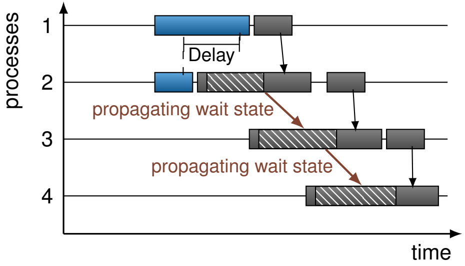
Wait-state classification

Distinguish propagating and terminal wait states



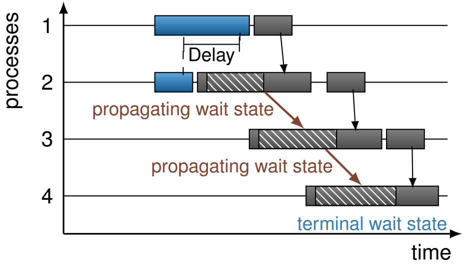
Wait-state classification

Distinguish propagating and terminal wait states



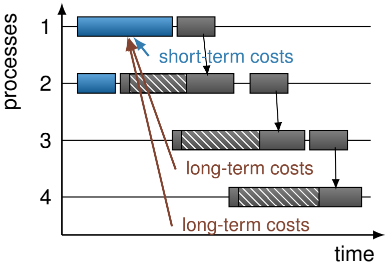
Wait-state classification

Distinguish propagating and terminal wait states



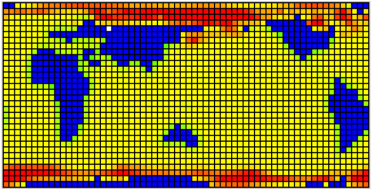
Assigning delay costs

- Delay costs represent amount of wait time caused by a delay
  - Short-term costs represent wait states caused directly
  - Long-term costs represent wait states caused via propagation



Case study: CESM sea ice model

- Analysis of imbalance in CESM sea ice model
- Performance data mapped onto application topology

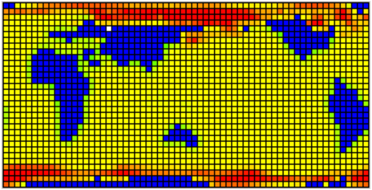


Distribution of computation time

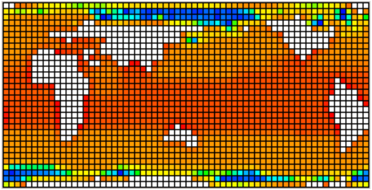
CICE setup: 2048 processes on BG/P, 1° dipole grid, cartesian grid decomposition

Case study: CESM sea ice model

- Analysis of imbalance in CESM sea ice model
- Performance data mapped onto application topology



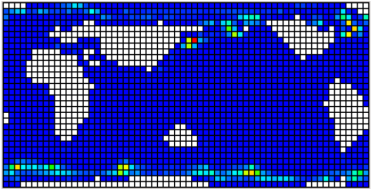
Distribution of computation time



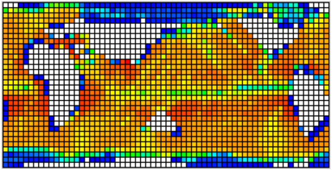
Distribution of late-sender waiting time

CICE setup: 2048 processes on BG/P, 1° dipole grid, cartesian grid decomposition

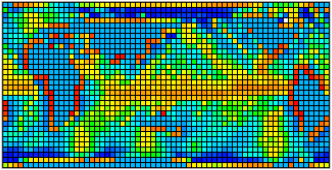
CESM sea ice model: wait-state formation



Distribution of delay costs



Propagating wait states



Terminal wait states

25% Short-term
75% Long-term

German Research School
for Simulation Sciences

Parallel Performance Analysis

Root-Cause Analysis

- Concepts
- Case study

Critical-Path Analysis

- The critical path
- Critical-path imbalance indicator
- Analysis of MPMD programs

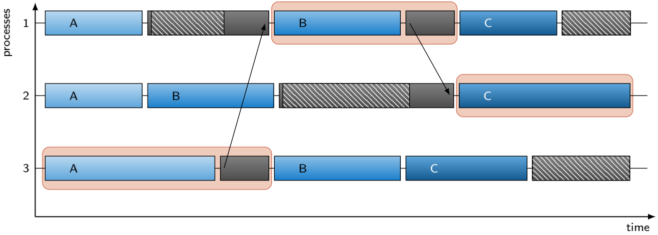
Implementation

- Parallel trace replay
- Scalability evaluation

JÜLICH FORSCHUNGSZENTRUM RWTH AACHEN UNIVERSITY

Critical-path analysis

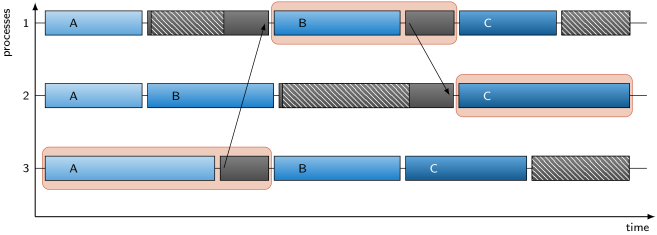
- Use automatic trace analysis to extract the critical path



Critical path in a parallel program (shown in red)

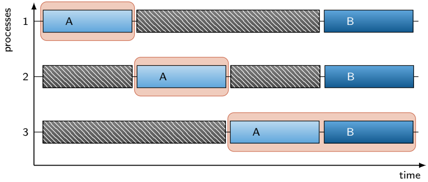
Critical-path analysis

- Use automatic trace analysis to extract the critical path
- Performance indicators show bottlenecks at single glance



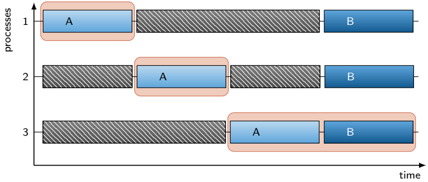
Critical path in a parallel program (shown in red)

Critical-path profile and imbalance



Timeline

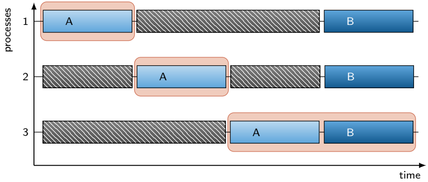
Critical-path profile and imbalance



Timeline

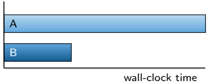
Summary profile

Critical-path profile and imbalance



Timeline

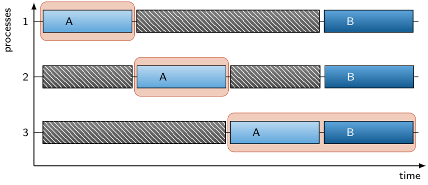
Summary profile



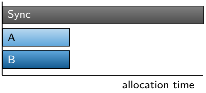
Critical-path profile

- Critical-path profile shows wall-clock time consumption

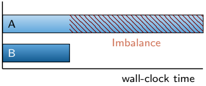
Critical-path profile and imbalance



Timeline



Summary profile

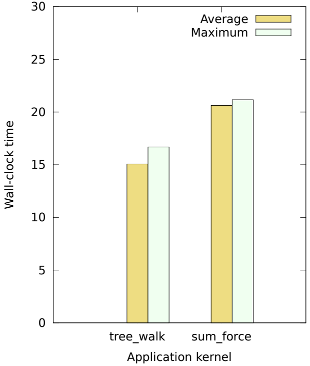


Critical-path profile

- Critical-path profile shows wall-clock time consumption
- Critical imbalance indicator finds inefficient parallelism
  - Imbalance $= T_{critical} - T_{avg}$

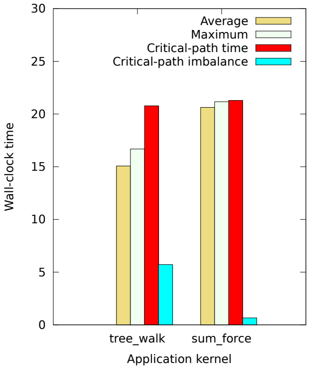
Example: PEPC

- Analysis of plasma-physics code PEPC using 512 processes on Blue Gene/P



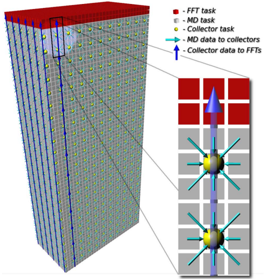
Example: PEPC

- Analysis of plasma-physics code PEPC using 512 processes on Blue Gene/P
- Profile metrics underestimate performance impact of `tree_walk` kernel due to dynamic load imbalance



Analysis of MPMD programs

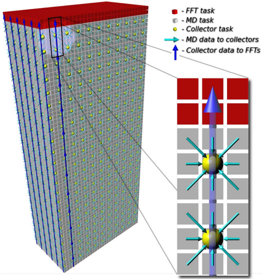
- Processes execute different activities
  - E.g. master-worker



Heterogeneous decomposition in ddcMDD.
Image from Richards et al.: Beyond Homogeneous Decomposition, SC'10

Analysis of MPMD programs

- Processes execute different activities
  - E.g. master-worker
- Complex imbalance analysis issues
  - Not supported by existing tools
  - Imbalance quantification needs to incorporate partition sizes
  - More knobs to tune

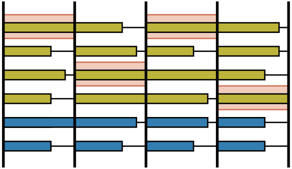


Heterogeneous decomposition in ddcMDD.
Image from Richards et al.: Beyond Homogeneous Decomposition, SC'10

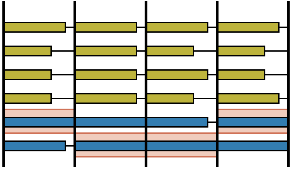
German Research School for Simulation Sciences

Performance impact indicators

- Denote allocation-time costs of imbalance
  - Map wait time onto critical-path activites with excess time
  - Distinguish intra-partition and inter-partition imbalance costs



High intra-partition costs, low inter-partition costs



Very high inter-partition costs, low intra-partition costs

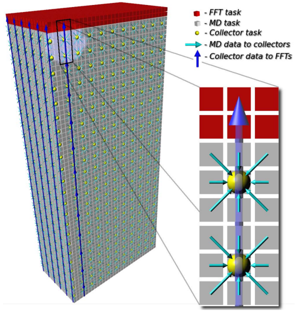
JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

German Research School
for Simulation Sciences

Example: ddcMD

ddcMD molecular dynamics simulation on Blue Gene/P

- *Particle* and *mesh* forces calculated in different partitions



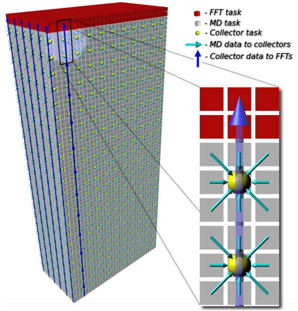
D. Richards et al.: Beyond Homogeneous Decomposition, SC'10

German Research School
for Simulation Sciences

Example: ddcMD

ddcMD molecular dynamics simulation on Blue Gene/P

- *Particle* and *mesh* forces calculated in different partitions
- Fixed partition sizes: 3840+256 processes



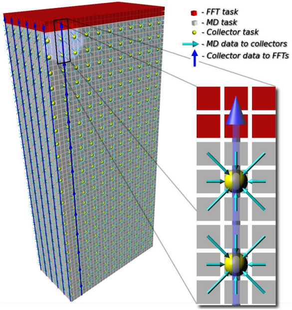
D. Richards et al.: Beyond Homogeneous Decomposition, SC’10

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Example: ddcMD

ddcMD molecular dynamics simulation on Blue Gene/P

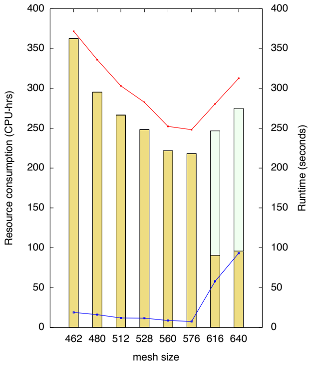
- *Particle* and *mesh* forces calculated in different partitions
- Fixed partition sizes: 3840+256 processes
- Tune mesh size to adjust load balance



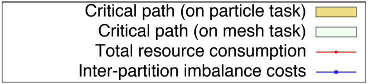
D. Richards et al.: Beyond Homogeneous Decomposition, SC'10

German Research School
for Simulation Sciences

ddcMD: mesh size tuning



- Small mesh size increases workload of particle tasks
- Increasing mesh size shifts critical path to mesh tasks



JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

German Research School
for Simulation Sciences

Parallel Performance Analysis

Root-Cause Analysis
- Concepts
- Case study

Critical-Path Analysis
- The critical path
- Critical-path imbalance indicator
- Analysis of MPMD programs

Implementation
- Parallel trace replay
- Scalability evaluation

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

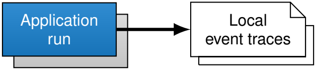
Implementation

- Integrated in Scalasca trace analysis toolset

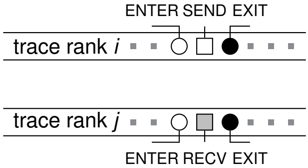
scalasca

- Highly scalable parallel trace analysis
  - So far only for wait-state detection

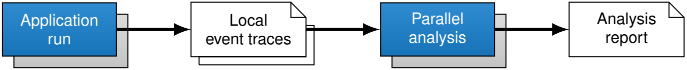
Parallel trace replay



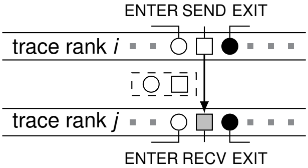
- Application records timestamped communication events
  - One trace file per process



Parallel trace replay

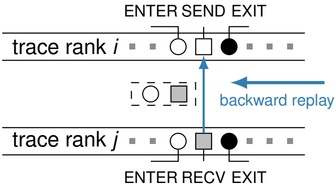


- Application records timestamped communication events
  - One trace file per process
- Analysis processes traverse traces in parallel
  - Exchange information at original synchronization points

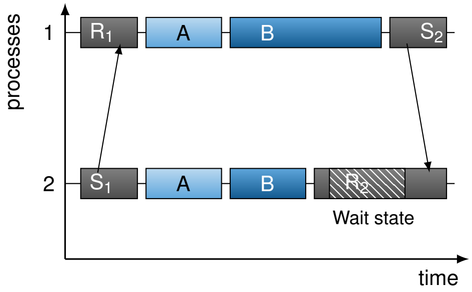


Trace analysis extensions

- Use multiple replay passes
- Backward replay lets data travel from effect to cause



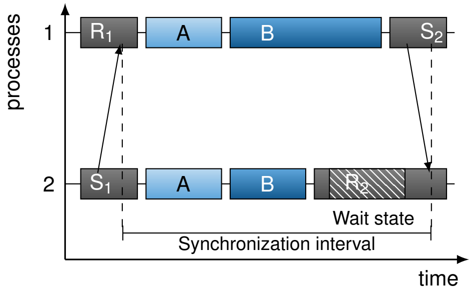
Delay detection via backward replay



German Research School for Simulation Sciences

Delay detection via backward replay

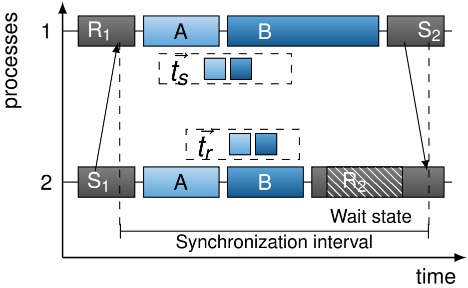
1. Identify synchronization interval



JÜLICH FORSCHUNGSZENTRUM RWTH AACHEN UNIVERSITY

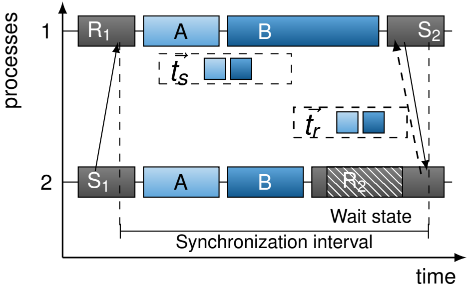
Delay detection via backward replay

1. Identify synchronization interval
2. Determine *time vectors* $\vec{t_s}$ and $\vec{t_r}$



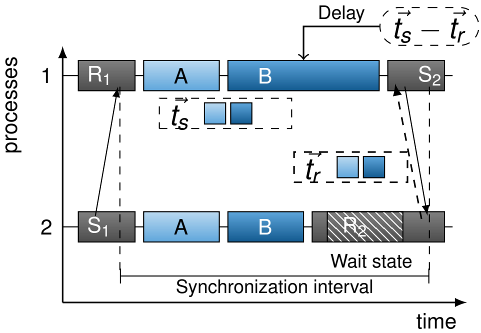
Delay detection via backward replay

1. Identify synchronization interval
2. Determine *time vectors* $\vec{t_s}$ and $\vec{t_r}$
3. Transfer time vector $\vec{t_r}$



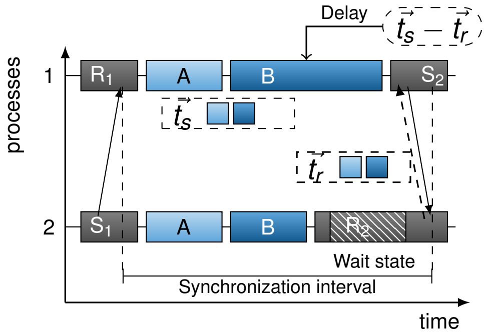
Delay detection via backward replay

1. Identify synchronization interval
2. Determine *time vectors* $\vec{t_s}$ and $\vec{t_r}$
3. Transfer time vector $\vec{t_r}$
4. Locate delay

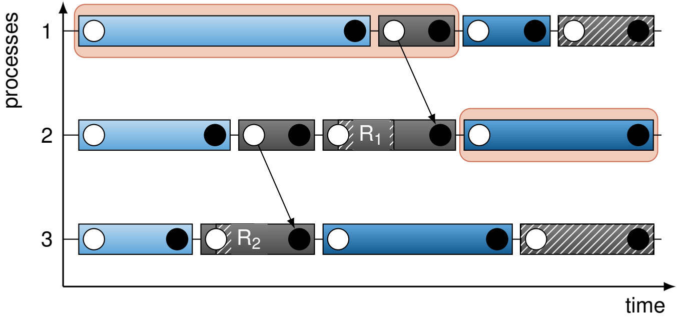


Delay detection via backward replay

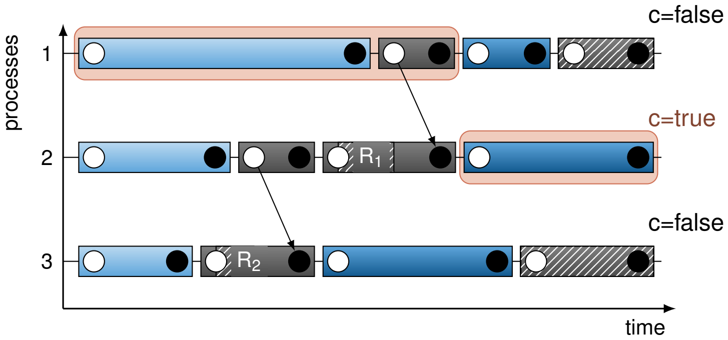
1. Identify synchronization interval
2. Determine *time vectors* $\vec{t_s}$ and $\vec{t_r}$
3. Transfer time vector $\vec{t_r}$
4. Locate delay
5. Calculate costs



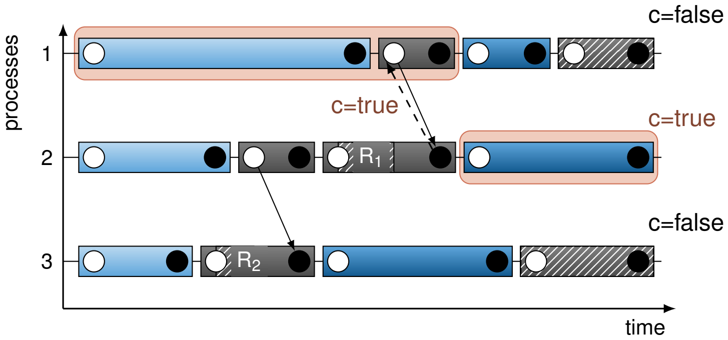
Critical-path extraction in backward replay



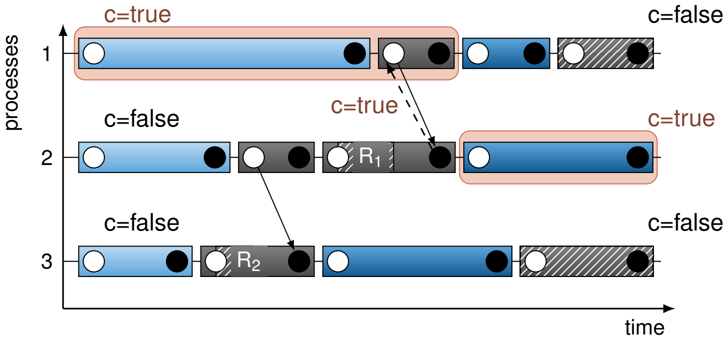
Critical-path extraction in backward replay



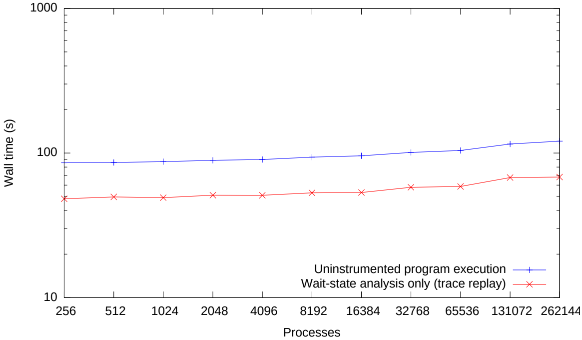
Critical-path extraction in backward replay



Critical-path extraction in backward replay

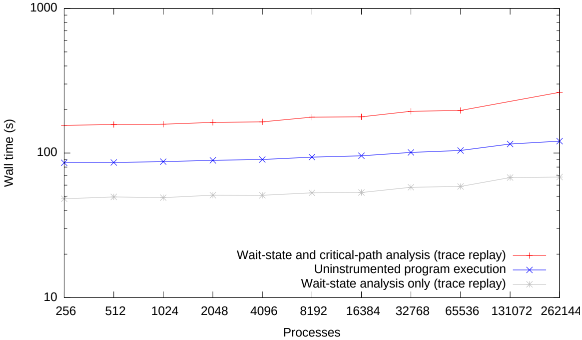


Scalability



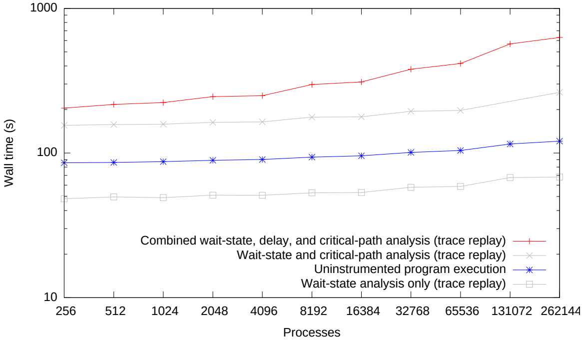
Scalability of root-cause and critical-path analysis for the Sweep3D benchmark on Blue Gene/P

Scalability



Scalability of root-cause and critical-path analysis for the Sweep3D benchmark on Blue Gene/P

Scalability



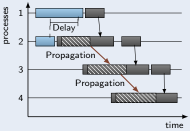
Scalability of root-cause and critical-path analysis for the Sweep3D benchmark on Blue Gene/P

German Research School for Simulation Sciences

Summary

Two novel methods to locate and quantify imbalance

Root-cause analysis



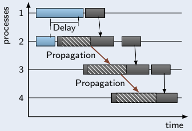
Identifies delays and explains formation of wait states

JÜLICH FORSCHUNGSZENTRUM RWTHAACHEN UNIVERSITY

Summary

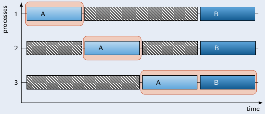
Two novel methods to locate and quantify imbalance

Root-cause analysis



Identifies delays and explains formation of wait states

Critical-path analysis

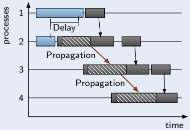


Determines impact of inefficiency on runtime

Summary

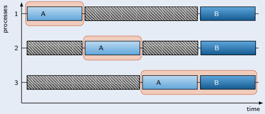
Two novel methods to locate and quantify imbalance

Root-cause analysis



Identifies delays and explains formation of wait states

Critical-path analysis



Determines impact of inefficiency on runtime

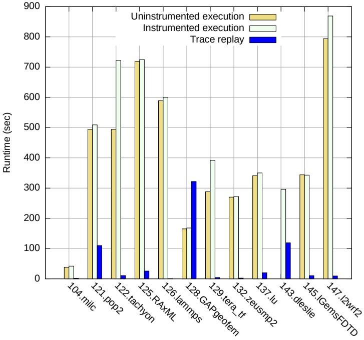
Highly scalable implementation for $\mathcal{O}(100k)$ processes

Thank you

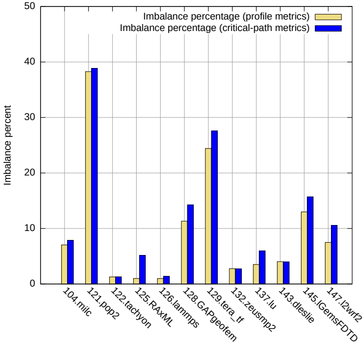
Further reading:

- D. Böhme, B. R. de Supinski, M. Geimer, M. Schulz, and F. Wolf: *Scalable Critical-Path Based Performance Analysis*. IPDPS 2012.
- D. Böhme, M. Geimer, and F. Wolf: *Characterizing Load and Communication Imbalance in Large-Scale Parallel Applications*. IPDPS PhD Forum 2012.
- D. Böhme, M. Geimer, F. Wolf, and L. Arnold: *Identifying the root causes of wait states in large-scale parallel applications*. ICPP 2010. Best paper award.
- D. Böhme, M.-A. Hermanns, M. Geimer, and F. Wolf: *Performance simulation of non-blocking communication in message-passing applications*. PROPER 2009.

SPECMPI case studies



Tracing run-time overhead and trace replay times



Critical-path vs. profile load imbalance metrics