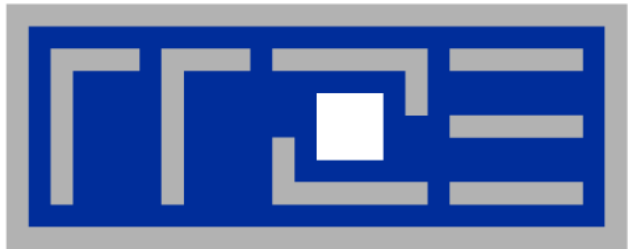
http://goo.gl/forms/hiXM5Feu3B



Code optimization war stories

Georg Hager, Jan Treibig
Erlangen Regional Computing Center (RRZE)
University of Erlangen-Nuremberg, Germany

SC14 BoF session
November 18, 2014
New Orleans, LA

Why are we here?

- **Share interesting code optimization ventures**
- **Interact with YOU (Yes. You.)**
- **Get feedback**

http://goo.gl/forms/hiXM5Feu3B

- **Lightning talks**
  - G. Hager: Optimizing a loop kernel from an FEM code – a study in compiler psychology
  - J. Treibig: Optimizing a large C++ numerical code
  - G. Hager: Best performance and energy for an MPI-parallel lattice-Boltzmann flow solver
  - T. William (TU Dresden): Analyzing an MD code – succeeding and failing
- **Wrap-up**
  - Please fill out our survey and take home the URL!

Some words about code optimization

What?

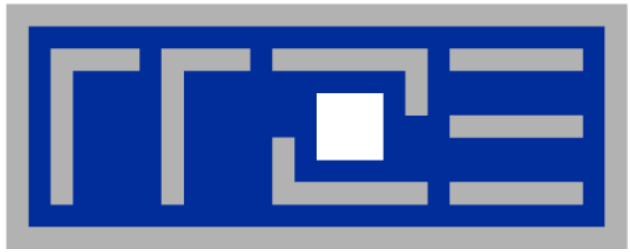
- Making code solve a problem faster
- Solve a larger problem in acceptable time
- Solve a problem with least amount of energy

How?

- Blindly apply code transformations
- Employ tools that tell me what‘s wrong
- Build performance models for guided optimizations

Why?

- Improved resource utilization
- Keep power envelope
- Observe energy budget



Optimizing a loop nest from an FEM code

Georg Hager

The original code (TU München, Chair for Computation in Engineering)

```
counter = 1
DO i=1, sizeA1
  DO j=1, i
    sum = 0.0D0
    DO k=1, 6
      sum = sum + A(k,i) * B(j,k)
    END DO
    C(counter) = C(counter) + sum
    counter = counter + 1
  END DO
END DO
```

P_{ser} = 4.5 Gflop/s

Compiler unrolls k loop completely but fails to vectorize j loop (dependency)

- sizeA1 = 600
- double precision arrays
- Loop nest executed many times
- Platform: Intel SNB @ 2.7 GHz (fixed)
- Intel compiler 13.1, **`-Ofast -xAVX`**

Step 1: Make counter variable non-counting

```
DO i=1, sizeA1
  counter = i*(i-1)/2
  DO j=1, i
    DO k=1, 6
      C(counter+j) = C(counter+j) + A(k,i) * B(j,k)
    END DO
  END DO
END DO
```

P_{ser} = 1.9 Gflop/s

- Compiler now vectorizes k loop, but does not unroll it completely
  → very inefficient vectorization with AVX

- Possible solution: Unroll k loop manually

Step 2: Manual unrolling, alignment (blatant lie!)

```
DO i=1, sizeA1
  counter = i*(i-1)/2
!DEC$ VECTOR ALIGNED
  DO j=1, i
    C(counter+j) = C(counter+j) + A(1,i) * B(j,1)
    C(counter+j) = C(counter+j) + A(2,i) * B(j,2)
    C(counter+j) = C(counter+j) + A(3,i) * B(j,3)
    C(counter+j) = C(counter+j) + A(4,i) * B(j,4)
    C(counter+j) = C(counter+j) + A(5,i) * B(j,5)
    C(counter+j) = C(counter+j) + A(6,i) * B(j,6)
  END DO
END DO
```

P_{ser} = 9.4 Gflop/s

- j loop now SIMD vectorized
- Nothing is actually aligned, but the code works with AVX
  - When compiled with SSE vectorization the code crashes → WHY?

Step 3: Inner loop unrolling

```
DO i=1, sizeA1
  counter = i*(i-1)/2
!DEC$ VECTOR ALIGNED
!DEC$ unroll(4)
  DO j=1, i
    C(counter+j) = C(counter+j) + A(1,i) * B(j,1)
    C(counter+j) = C(counter+j) + A(2,i) * B(j,2)
    C(counter+j) = C(counter+j) + A(3,i) * B(j,3)
    C(counter+j) = C(counter+j) + A(4,i) * B(j,4)
    C(counter+j) = C(counter+j) + A(5,i) * B(j,5)
    C(counter+j) = C(counter+j) + A(6,i) * B(j,6)
  END DO
END DO
```

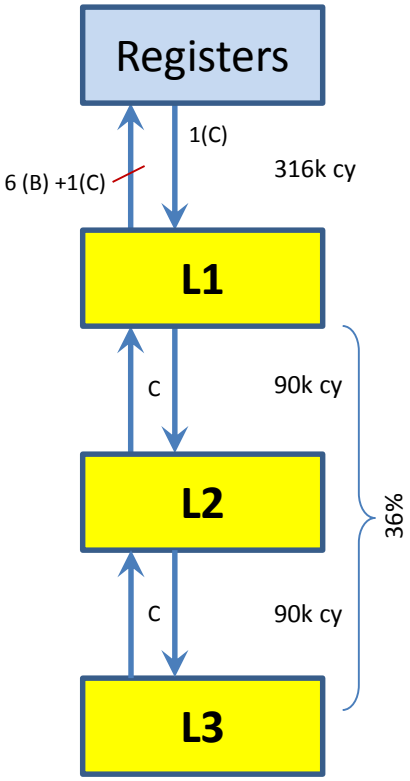
P_{ser} = 9.6 Gflop/s

- But is this good enough? Peak performance is 21.6 GFlop/s!?!??
  → Employ the ECM model!

  H. Stengel, J. Treibig, G. Hager, and G. Wellein: *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*. Preprint: arXiv:1410.5010

ECM Model

- Assumption: C comes from L3 (streaming), A and B are in L1 or registers




```
DO i=1, sizeA1
  counter = i*(i-1)/2
  DO j=1, i
    C(counter+j) = C(counter+j) + A(1,i) * B(j,1)
    C(counter+j) = C(counter+j) + A(2,i) * B(j,2)
    C(counter+j) = C(counter+j) + A(3,i) * B(j,3)
    C(counter+j) = C(counter+j) + A(4,i) * B(j,4)
    C(counter+j) = C(counter+j) + A(5,i) * B(j,5)
    C(counter+j) = C(counter+j) + A(6,i) * B(j,6)
  END DO
END DO
```

- L1: 7 LOADs, 1 STORE, 6 MULT, 6 ADD
  → LOAD limited → 48 Flops in 7 cycles
  → 18.5 Gflop/s
  → All work from L1: 600*601/2*7/4 cy = 316000 cy
- L2/L3: 600*601/2*8*2/32 cy = 90150 cy
- → P = 11.8 Gflop/s

ECM Model: Discrepancies

- Model: 11.8 Gflop/s, Measurement: 9.6 Gflop/s → 82%
- Possible explanations
  - Traffic for B from L2 not negligible
  - Non-streaming access to B in L2 (latency-bound?)
- Validation: Measure L2 cache traffic with likwid-perfctr (10000 kernel invocations)

```
$ likwid-perfctr -C N:1 -g L2 ./matrixMatrixProduct
```

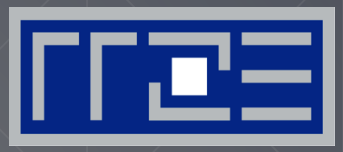
L2 Load [MBytes/s]	8371.3	
L2 Evict [MBytes/s]	6407.81	
L2 bandwidth [MBytes/s]	14779.1	
L2 data volume [GBytes]	33.4015	

Expected for C: Load == Evict

Expected for C: 28.8 GB

- Conclusion: Extra traffic must be caused by reloads of B from L2 cache (500 kB ≈ 20x reload)
- Explains ≈30% of the deviation

ERLANGEN REGIONAL COMPUTING CENTER



UTBEST3D – Optimizing a large object oriented numerical application

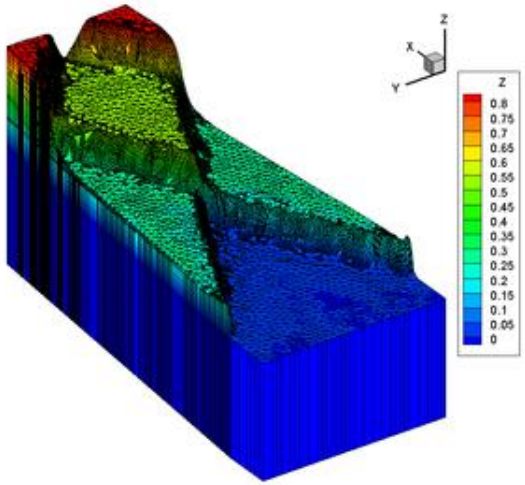
J. Treibig

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Introduction

- 3D shallow water code (Regional ocean model)
- Numerical method: **local discontinuous Galerkin**
- **Modular** (hierarchical) physical model from barotropic quasi-2D to full baroclinic model with turbulence closure: **varying number of equations and unknowns**
- **Equation types**: momentum, elevation, continuity, transport
- 3D mesh with **columns of prisms** and aligned 2D **triangular mesh**
- C++, OpenMP, MPI

University of Texas Bays and Estuaries Simulator



Problems with real (object oriented) application codes

C++ specific issues:

- Performance relies on proper function inlining

- Complex data structures:
  - Array of structures
  - Pointer chasing (Load traces)

- Large functionality causes control flow problems:
  - Create code variants for static control flow
  - Templates might help
  - Not much support from the programming languages

Hardware Performance Monitoring Profiling

Instruction type	% of overall instructions
LOAD/STORE	52.7%
BRANCH	**7.7%**
ARITHMETIC	26.3%
OTHER	**13.3%**

	1 thread	10 threads
MEM (GBytes/s)	1.05	**9.75**
Data (GBytes)	15.7	14.4
MFlops/s	1.09E+03	1.11E+4
Walltime	15.0 s	1.47 s

Reduce instruction work to enable better use of bandwidth

Optimization Steps

1. Make **nested loops more transparent**
   (work directly on data structures, make loop index visible) **15 s** to **14 s**, some packed SSE (~2%)
2. Exploit static information
   (local number of DOFs, number of transport unknowns) **14 s to 10 s**, no vectorization
3. Bring back **flexibility** (variable approx. orders, DOF iterators) **10s to 12s**
4. Eliminate custom **iterators 12s** to **9.5s**
5. Storage of some **precomputed values**
   (weights, depth, reorganized some indirections, ...) **9.5s** to **8.8s**
6. Evaluate variables for all quadrature points at once **8.8s** to **8.1s**
7. **Total walltime:** 159 s to 151 s

Dead Ends (or the struggle for SIMD)

- Eliminate quadrature by precomputed reference blocks (50% more flops in theory and much more data!)
- Other data structures for entity data (depth, forcings, ...)
- Quadrature- and prism-loops swapped (20% of arithmetic SSE-packed, 40% slower)
- Separate loops for evaluation and RHS-update (no SSE, 25% slower)
- I #pragma simd (vectorized – but same walltime)

Hardware Performance Monitoring Profiling

Instruction type	OLD	NEW
LOAD/STORE	52.7%	56.6%
BRANCH	7.7%	**2.0%**
ARITHMETIC	26.3%	37.5%
OTHER	13.3%	**3.9%**

	OLD		NEW	
	1 thread	10 threads	1 thread	10 threads
MEM (GBytes/s)	1.05	9.75	1.71	**15.7**
Data (GBytes)	15.7	14.4	14.8	13.9
MFlops/s	1.09E+03	1.11E+4	1.73E+03	1.69E+04
Walltime	15.0 s	1.47 s	8.67 s	0.88 s

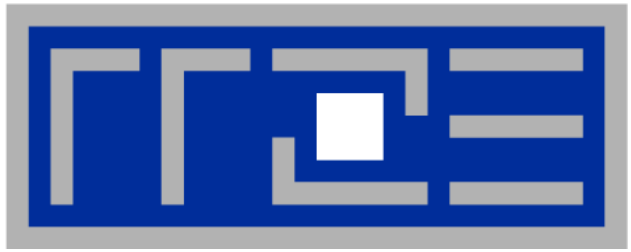
Conclusion

- Large Scale C++ application often suffer from language induced instruction overhead.

- Target of this effort was to reduce this overhead and make better use of available resources (memory bandwidth and SIMD)

- We **failed to leverage SIMD** for this application
- Memory bandwidth is still **not saturated**
- To apply these optimizations to all kernels is a **huge effort**

- Still we got the a **50% improvement** for the kernels we looked at



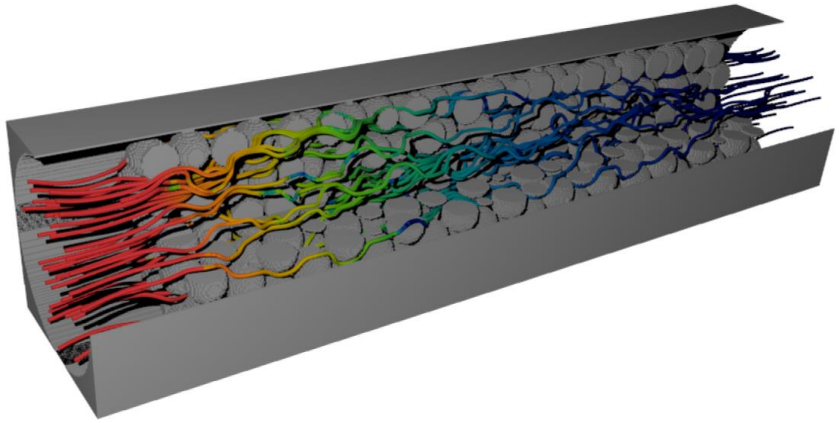
Getting lowest energy to solution for a highly parallel lattice-Boltzmann flow solver

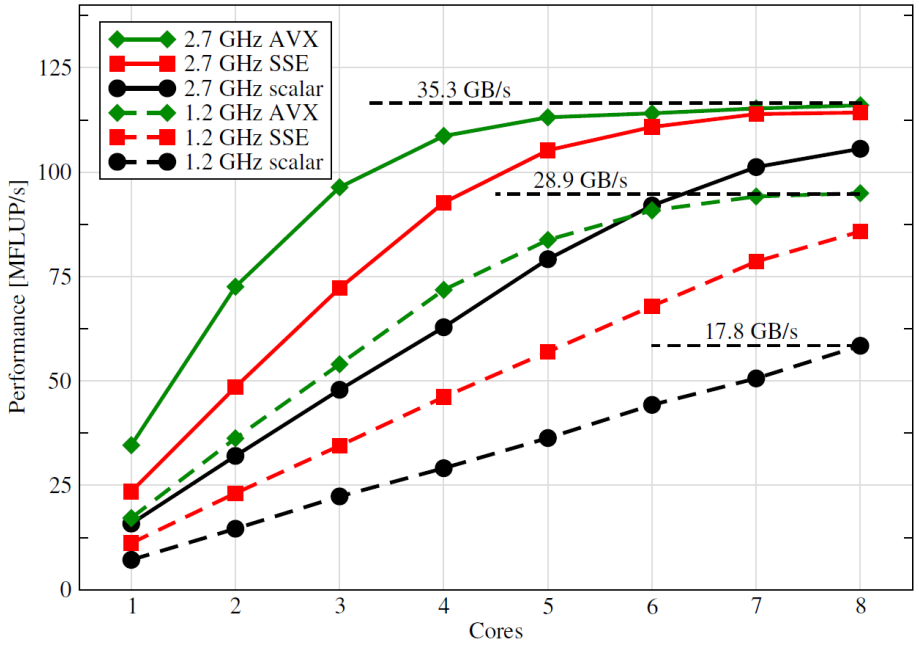
Georg Hager

M. Wittmann, G. Hager, T. Zeiser, J. Treibig, and G. Wellein: *Chip-level and multi-node analysis of energy-optimized lattice-Boltzmann CFD simulations*. Submitted.
Preprint: arXiv:1304.7664

ILBDC Code

- Sparse representation lattice-Boltzmann flow solver
- Well suited for highly porous geometries, MPI parallel
- „AA pattern“ propagation → high computational intensity
  - 304 bytes/LUP for even time step
  - 376 bytes/LUP for odd time step
- Saturating performance for vectorized code on one Intel Sandy Bridge socket





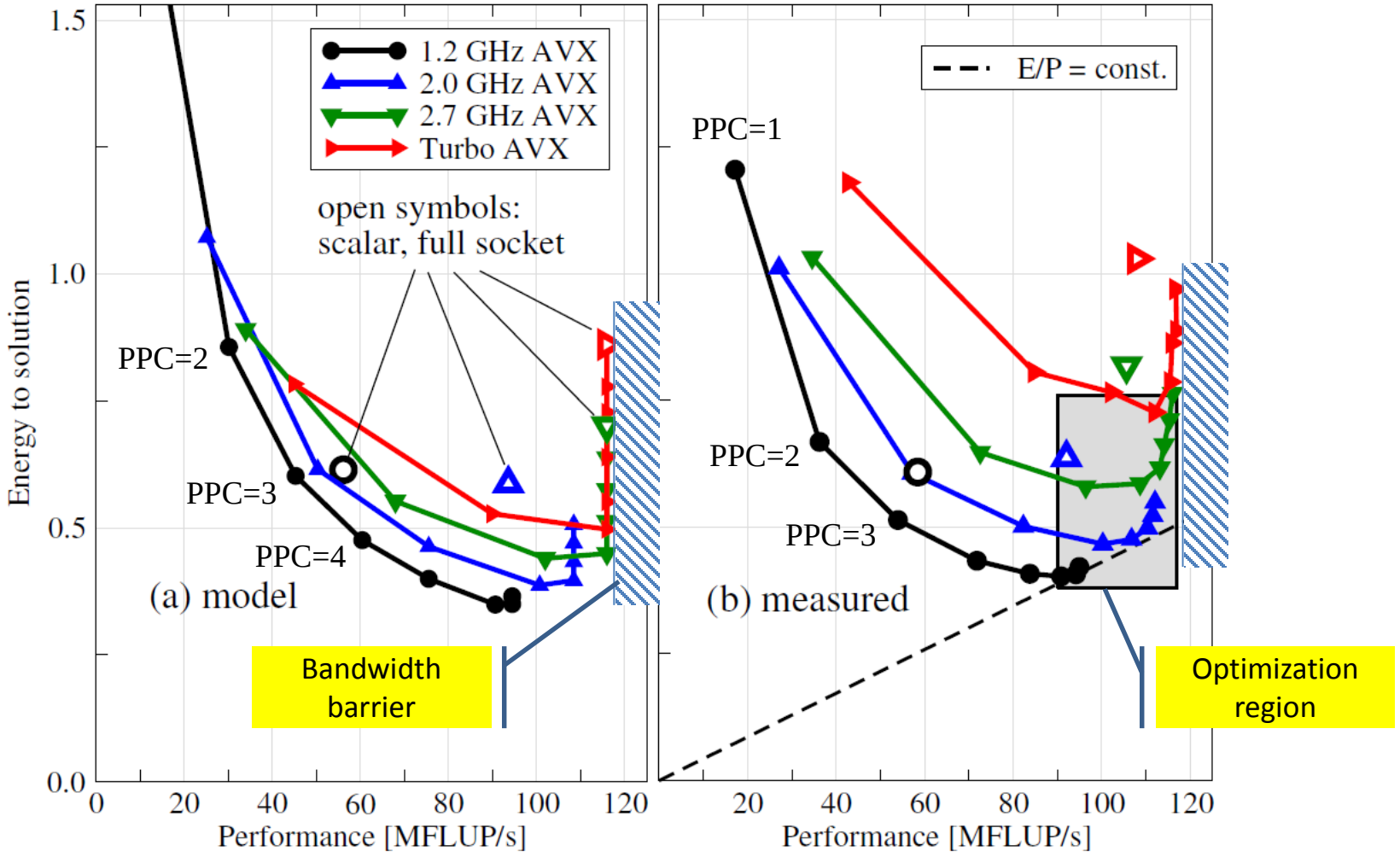
Energy to solution model

$$E = \frac{W_0 + W_2 f^2 n}{\min\left(\frac{f}{f_0} n P_0, P_{\max}\right)}$$

f:	CPU clock speed
f_0:	CPU base clock speed
n:	# of active cores
P_0:	single-thread performance
P_{max}:	saturated socket performance
W_0, W_1:	parameters (fixed by fitting)

Energy to solution vs. performance on the socket (SNB)

Model vs. Measurement at different clock speeds (PPC=proc.s per chip)



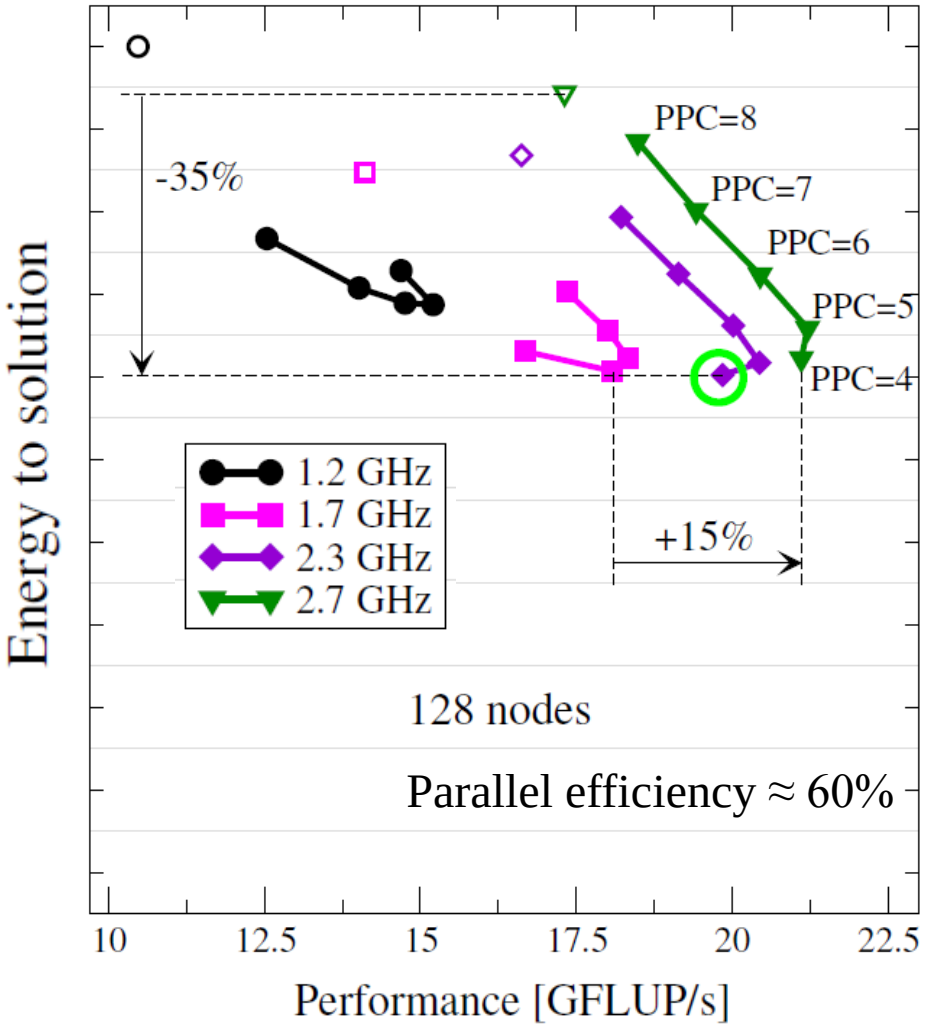
Energy to solution vs. performance on the cluster (SuperMUC)

How does that change when going multi-node with substantial communication overhead?

- Dependence on socket-level concurrency?
- Dependence on clock speed?

Observations

- Optimal PPC is crucial for lowest energy!
- Higher clock speed yields better performance without energy penalty!



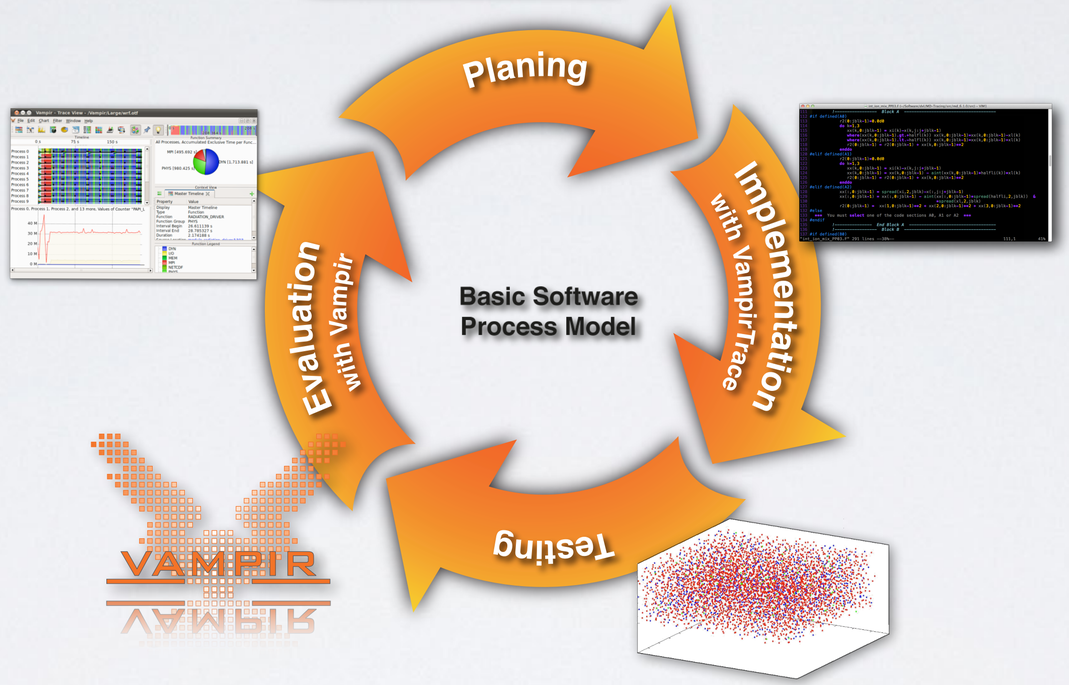
MOLECULAR DYNAMICS CODE Analysis

Evaluating Serial, Thread and Process-Parallel Performance

Molecular Dynamics

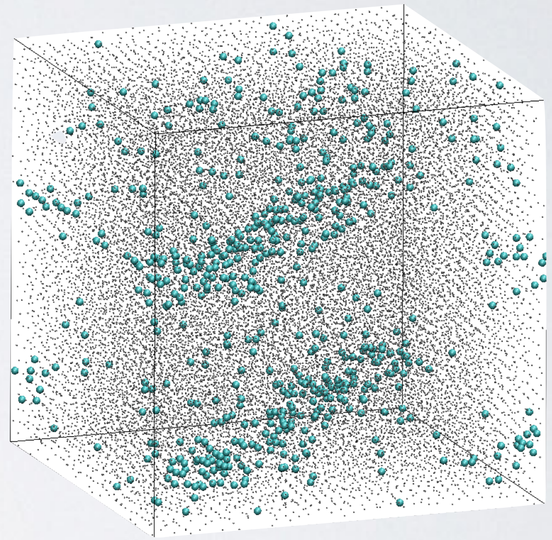
A molecular dynamics code simulating the diffusion in dense nuclear matter in white dwarf stars is analyzed. The code is highly configurable allowing MPI, OpenMP, or hybrid runs and additional fine tuning with a range of parameters.

$$V_c(r_{ij}) = \frac{Z_i Z_j e^2}{r_{ij}^2} \exp\left(-\frac{r_{ij}}{\lambda}\right)$$



The Idea

- Identify the best parameter set
- Use those candidate for further parallel analysis
- Measure the scalability limits
- Detect bottlenecks



Thomas William (ZIH, TU-Dresden, Germany)
Email: thomas.william@tu-dresden.de

This document was developed with support from the National Science Foundation (NSF) under Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Test-bed." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SCOPE OF THE CODE ANALYSIS

Hardware Details & Parameter Sweep

Measurement System:

- Cray XT5m
- 84 nodes with 672 cores
- 2 2.4 GHZ CPUs per node
- Quad-Core AMD Opteron(tm) Processor 23 (C2)

Time estimate per measurements and particles:

5k	~ 300 seconds	~5 minutes
27k	~ 4000 seconds	~1 h
55k	~ 35000 seconds	~10 h

Serial analysis:

- 55k particles
- Comparing optimization flags:
  - O2
  - O3
  - fastsse

Particle-type
- nucleon-nucleon
- ion pure
- ion mix

Input parameter
- simulation type
- number of particles
- number of time steps

Compiler Flags
- O2
- O3
- SSE

Loop combination
- Block A: A0-A2
- Block B: B1-B7
- Code-blocking: NBS

Code version
- Original: PP01
- Production: PP02
- Research: PP03

Parallelism
- serial: md
- OpenMP: md_omp
- MPI: md_mpi
- Hybrid: md_mpi_omp

hundreds of combinations

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

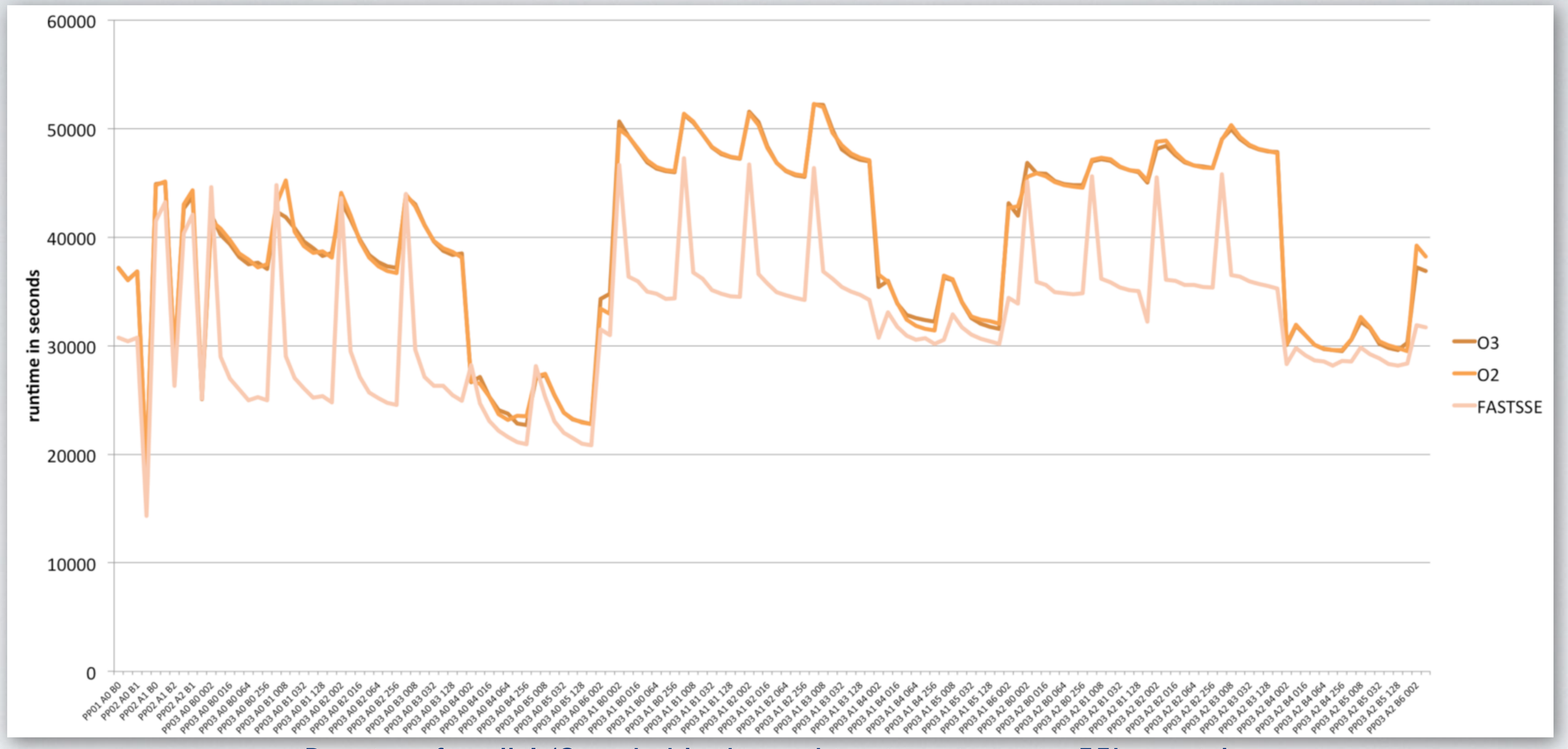
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS

Identifying The Best Configuration For Parallel Runs



Runtime for all 143 code block combinations, ion mix, 55k particles

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

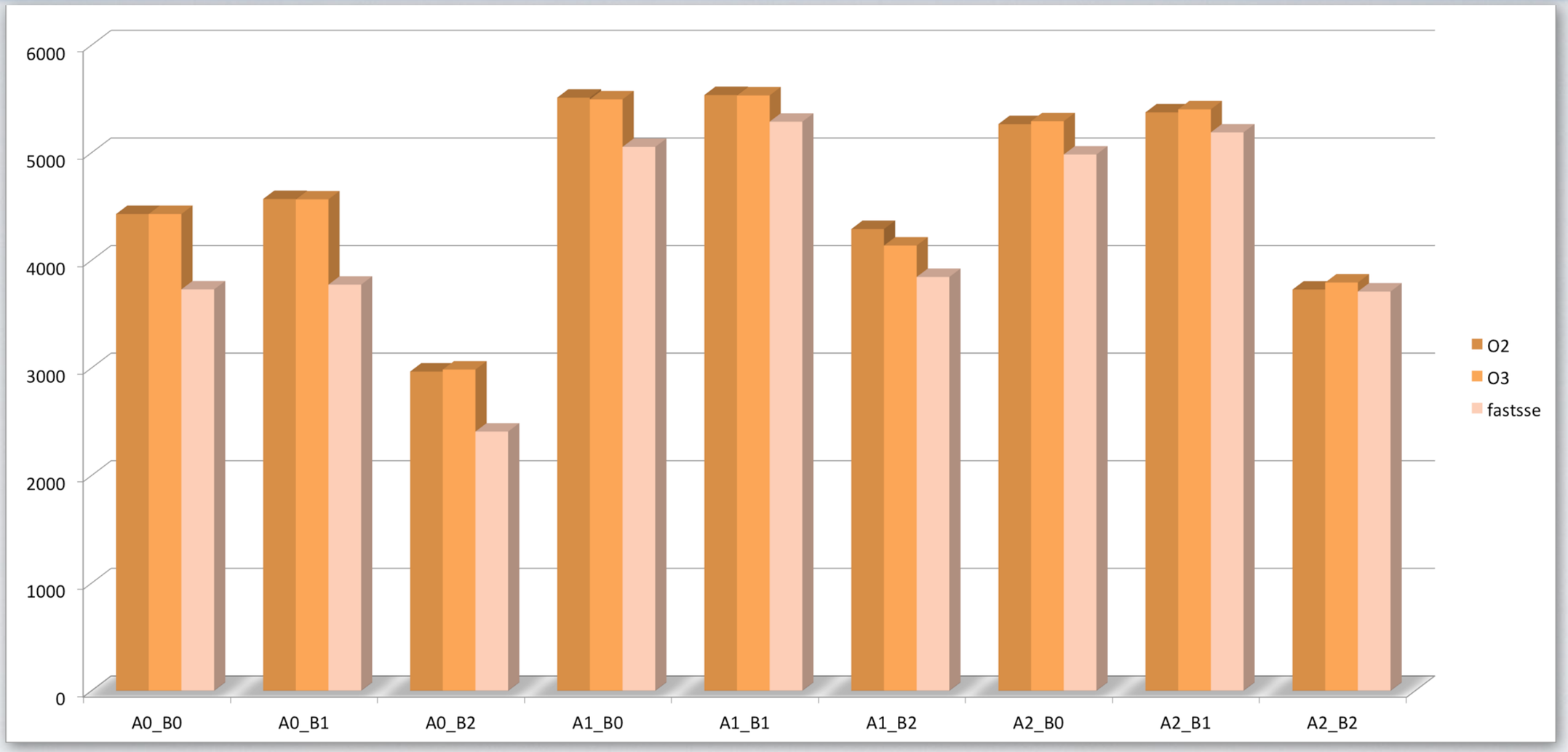
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS

Identifying Best Configuration For Parallel Runs



Runtime in seconds

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

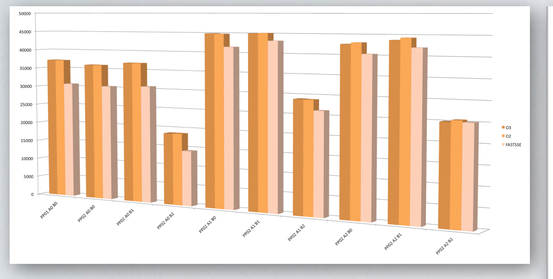
IU Bloomington
Pervasive Technology Institute

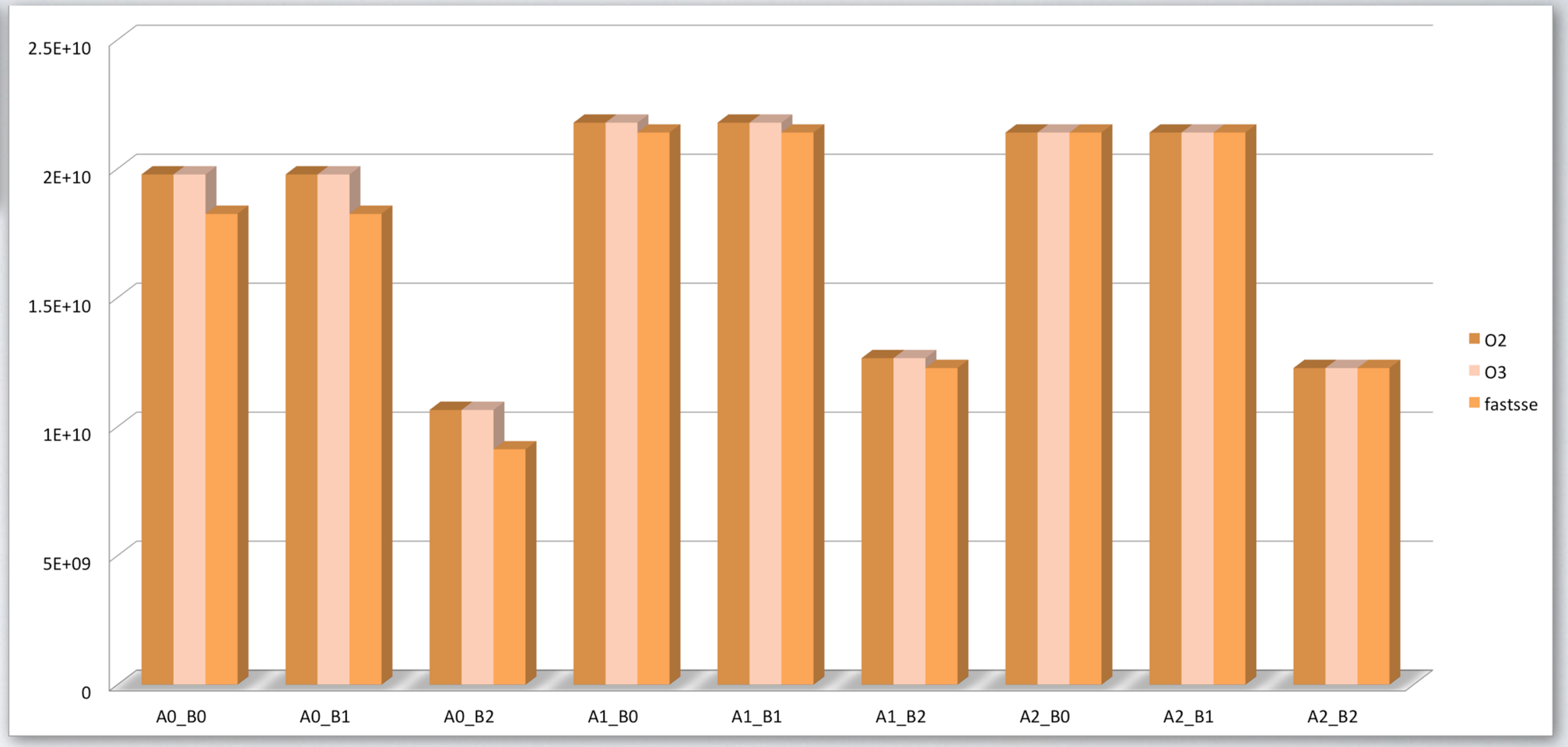
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS
PAPI Aided Analysis of PP02 Code Versions





PAPI_FP_INS

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

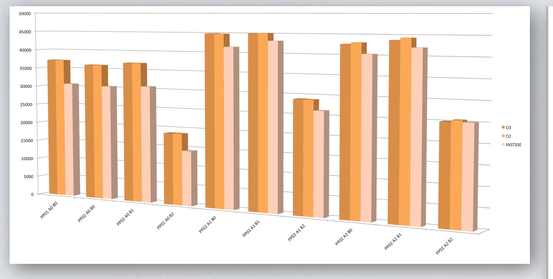
IU Bloomington
Pervasive Technology Institute

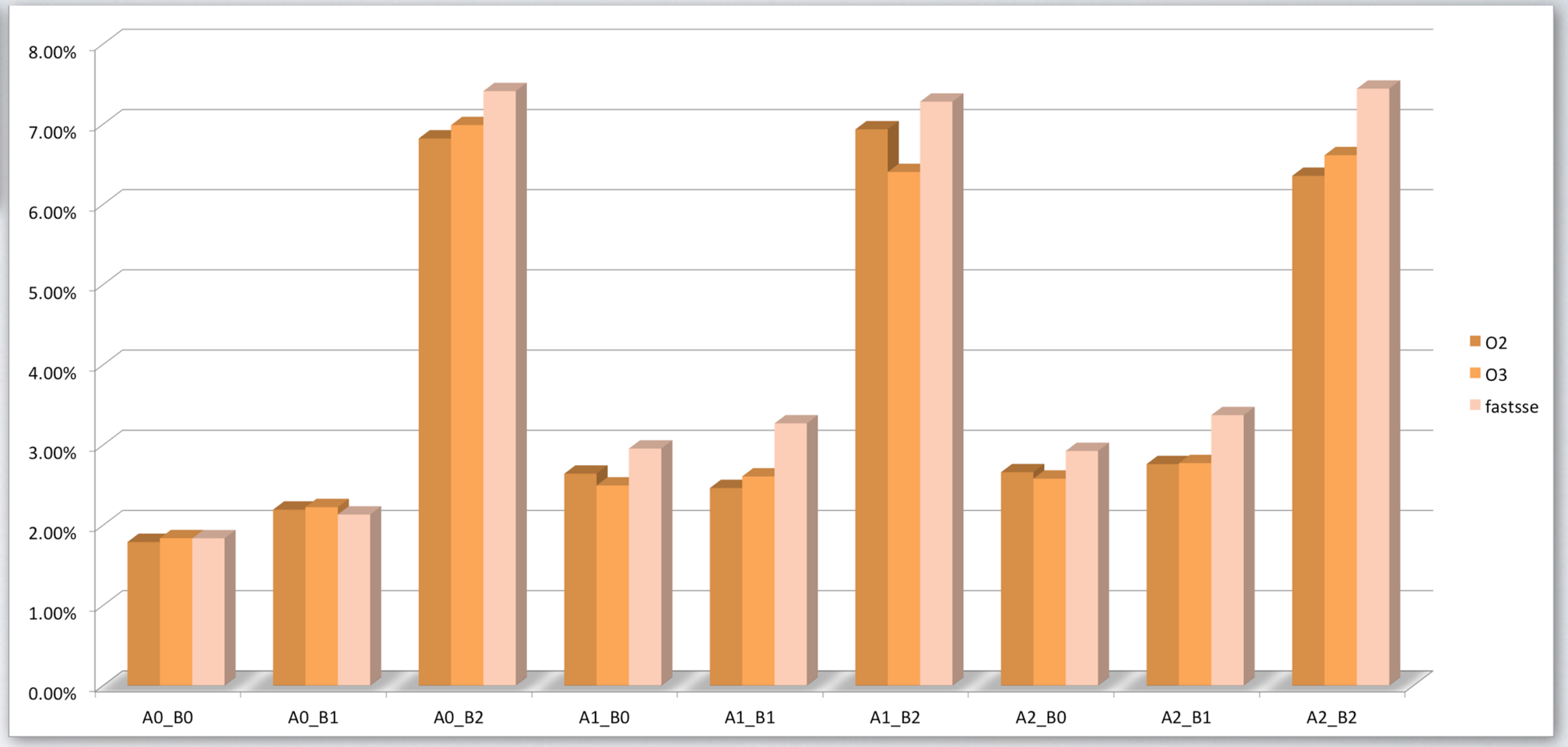
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS
PAPI Aided Analysis of PP02 Code Versions





FPU Idle

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

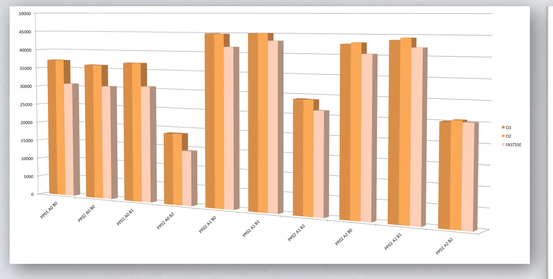
IU Bloomington
Pervasive Technology Institute

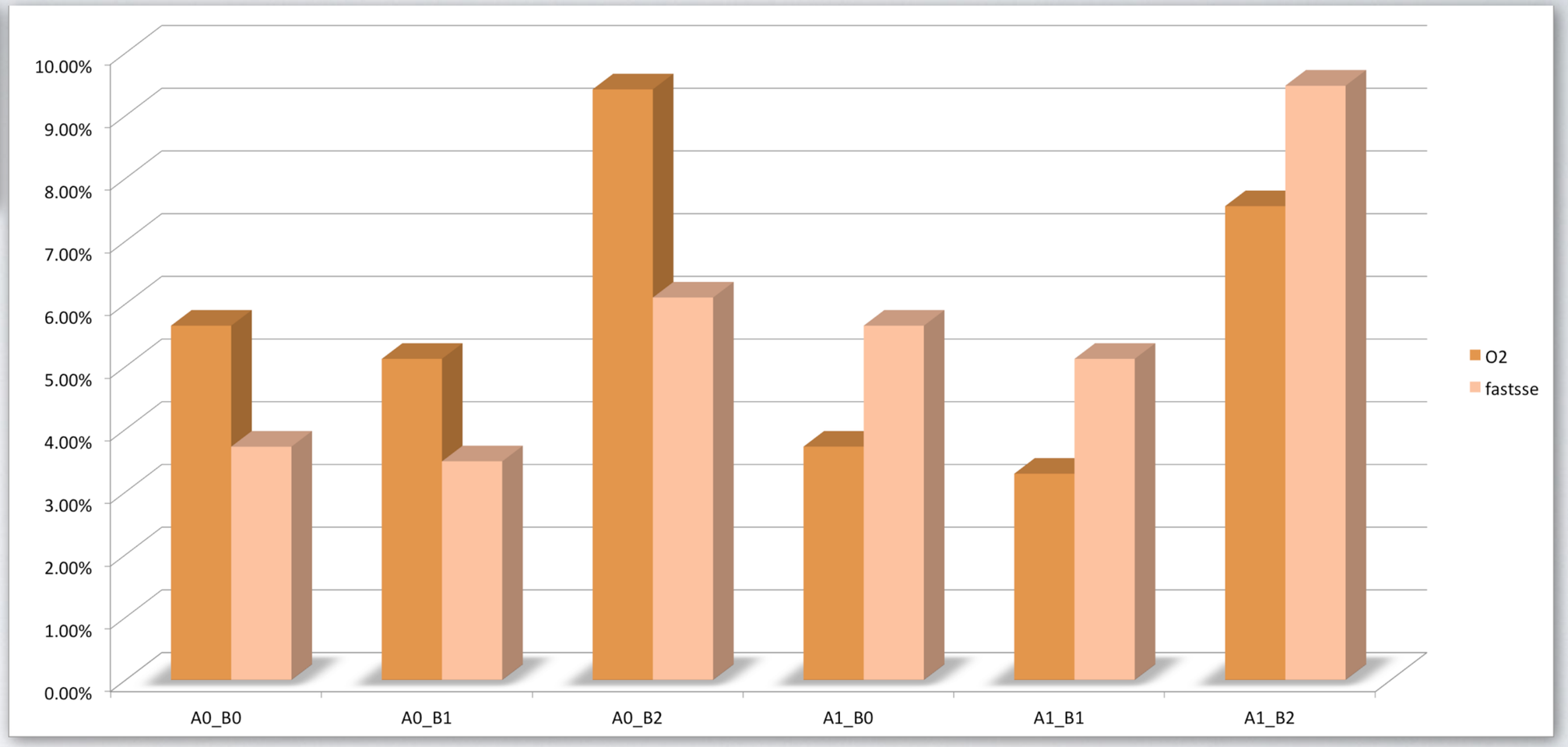
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS
PAPI Aided Analysis of PP02 Code Versions





Branches mispredicted

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

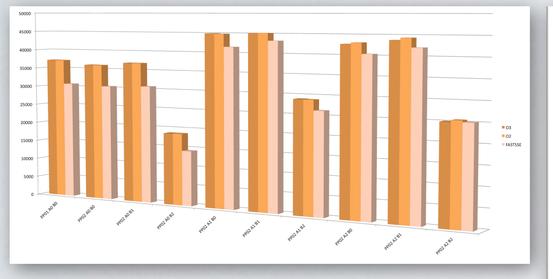
IU Bloomington
Pervasive Technology Institute

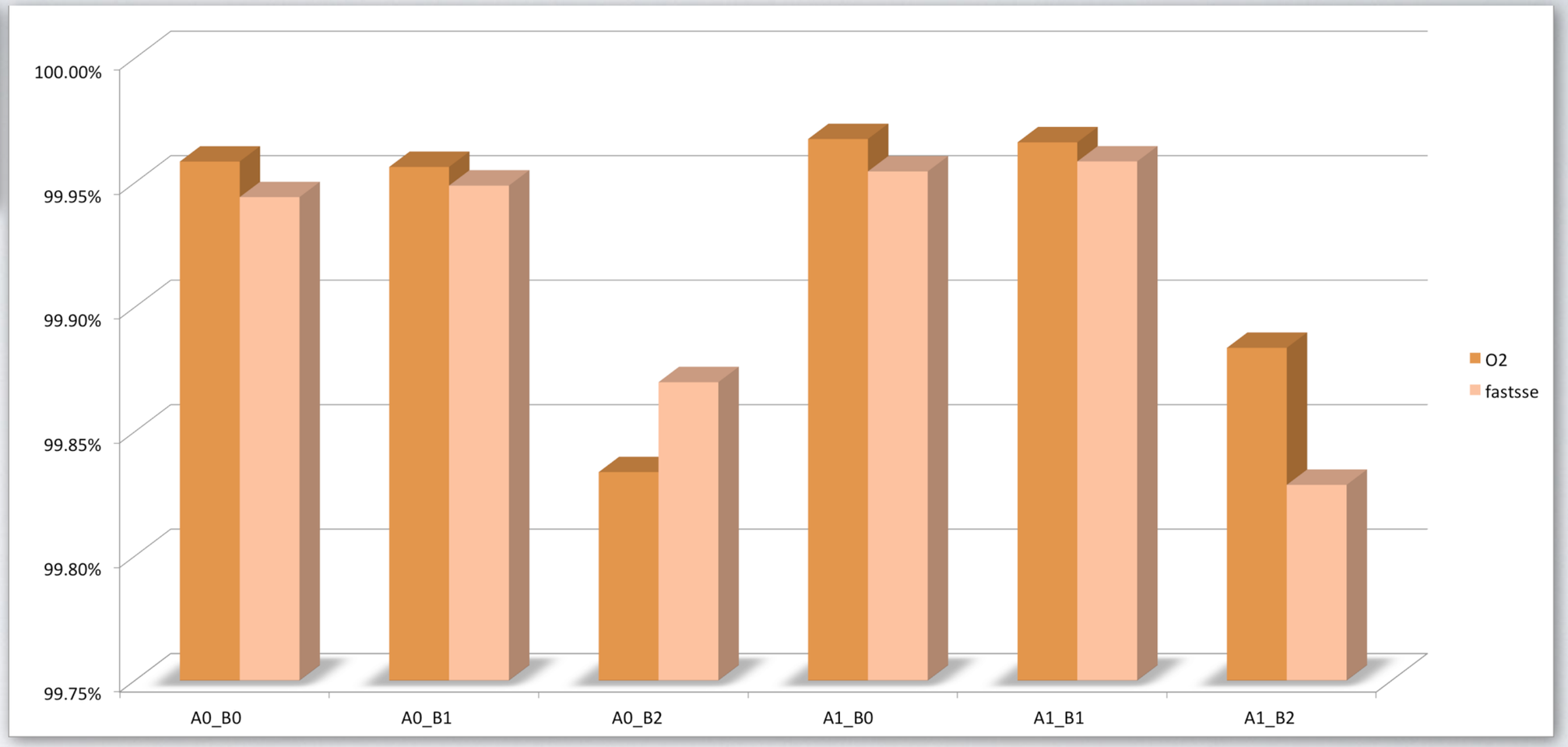
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS
PAPI Aided Analysis of PP02 Code Versions





L1 hit ratio

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

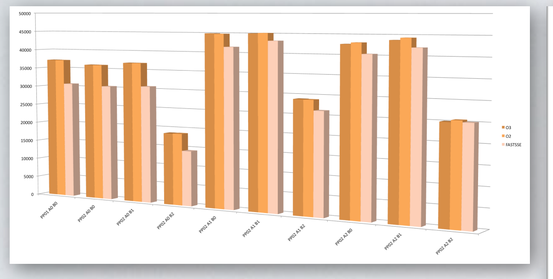
IU Bloomington
Pervasive Technology Institute

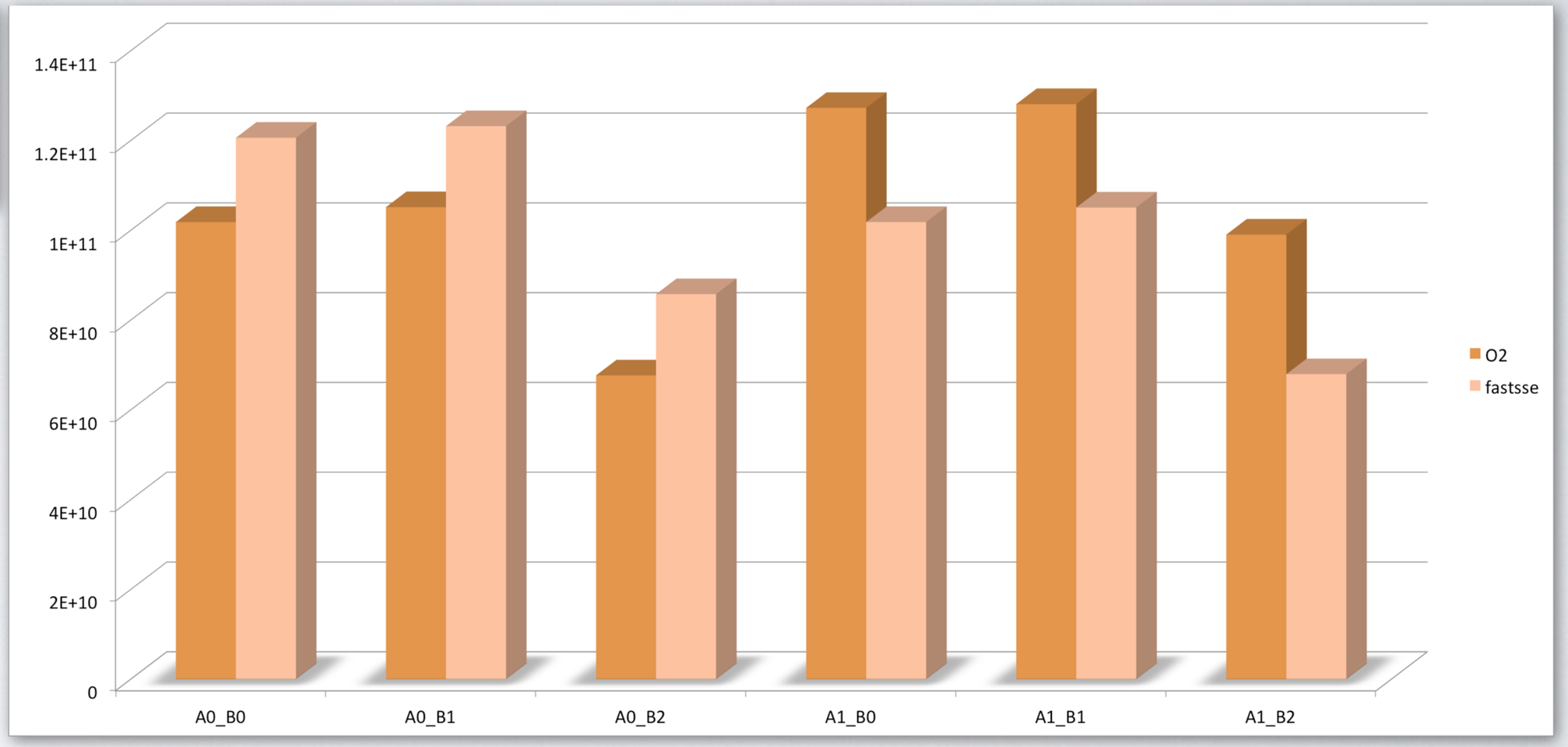
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SERIAL ANALYSIS
PAPI Aided Analysis of PP02 Code Versions





PAPI_TO_CYC

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

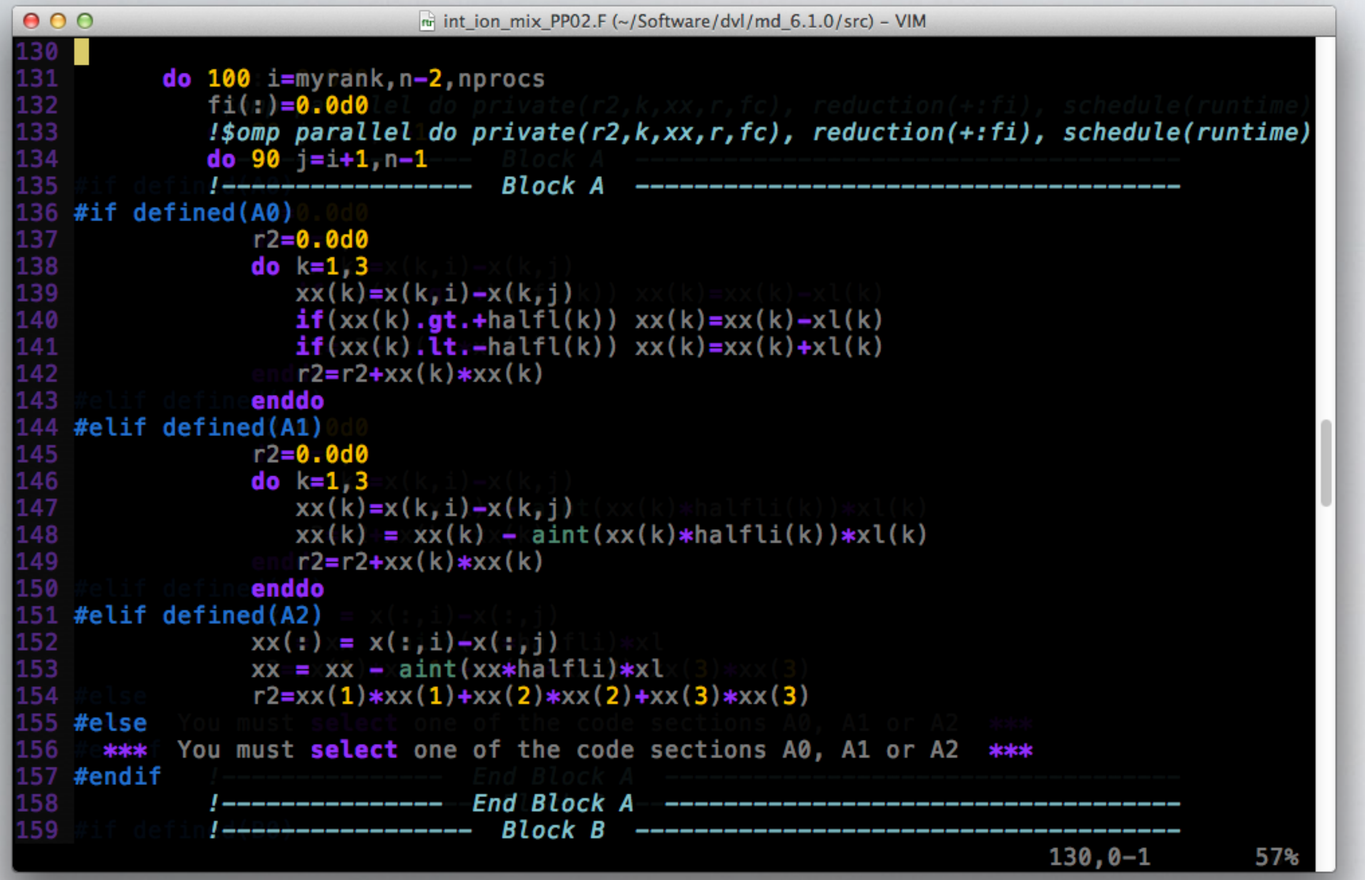
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SOURCE CODE ANALYSIS

Codeblocks Ax & Bx For Ion-Mix (PP02)



A0:

Branching

A1:

Arithmetic

A2:

Array syntax

Block A

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

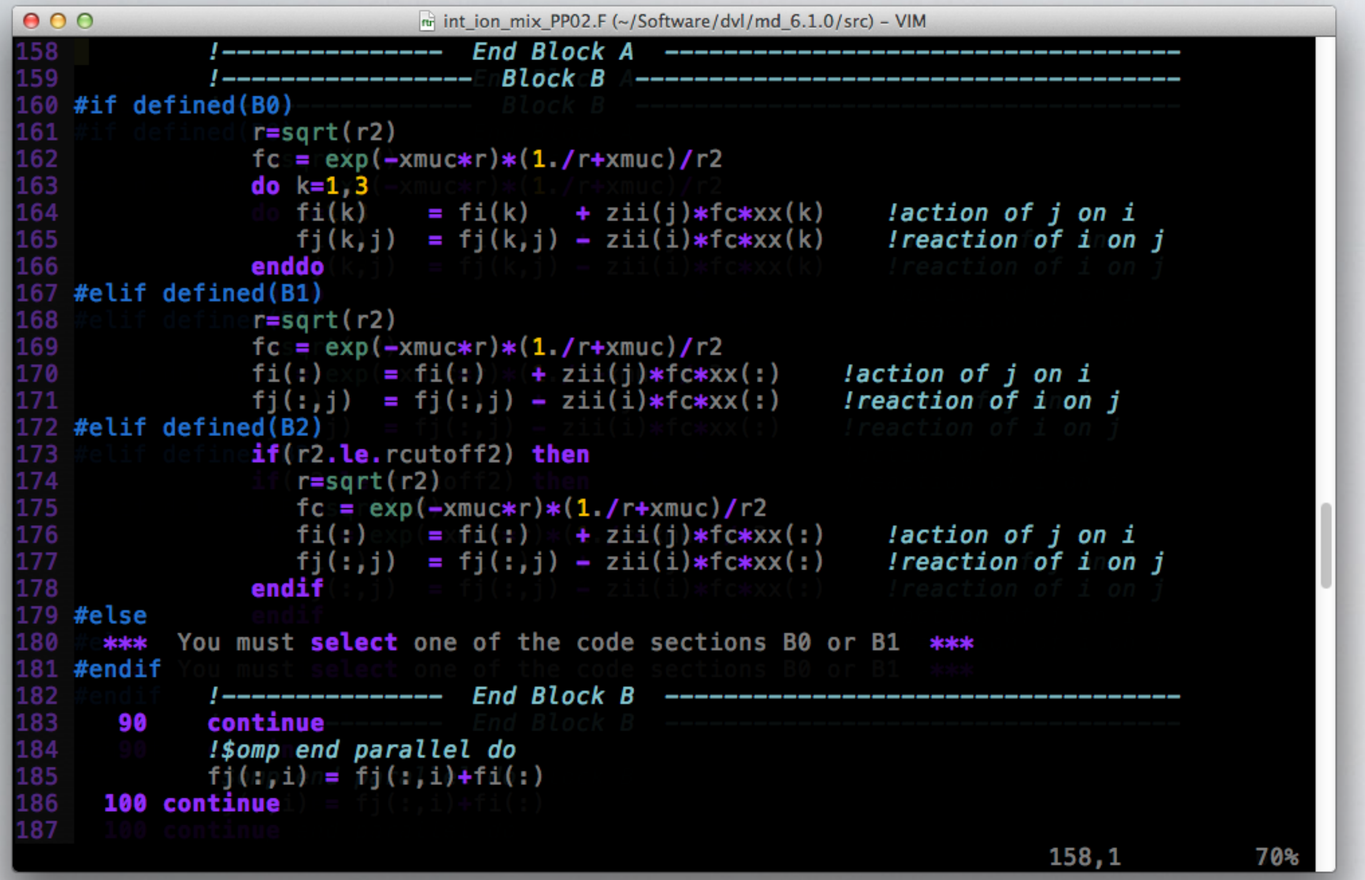
2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SOURCE CODE ANALYSIS

Codeblocks Ax & Bx For Ion-Mix (PP02)



Block B

B0:

Loop

B1:

No Cut-off Sphere

B2:

Cut-off Sphere

Result: **"Calculation does not beat branching"**

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

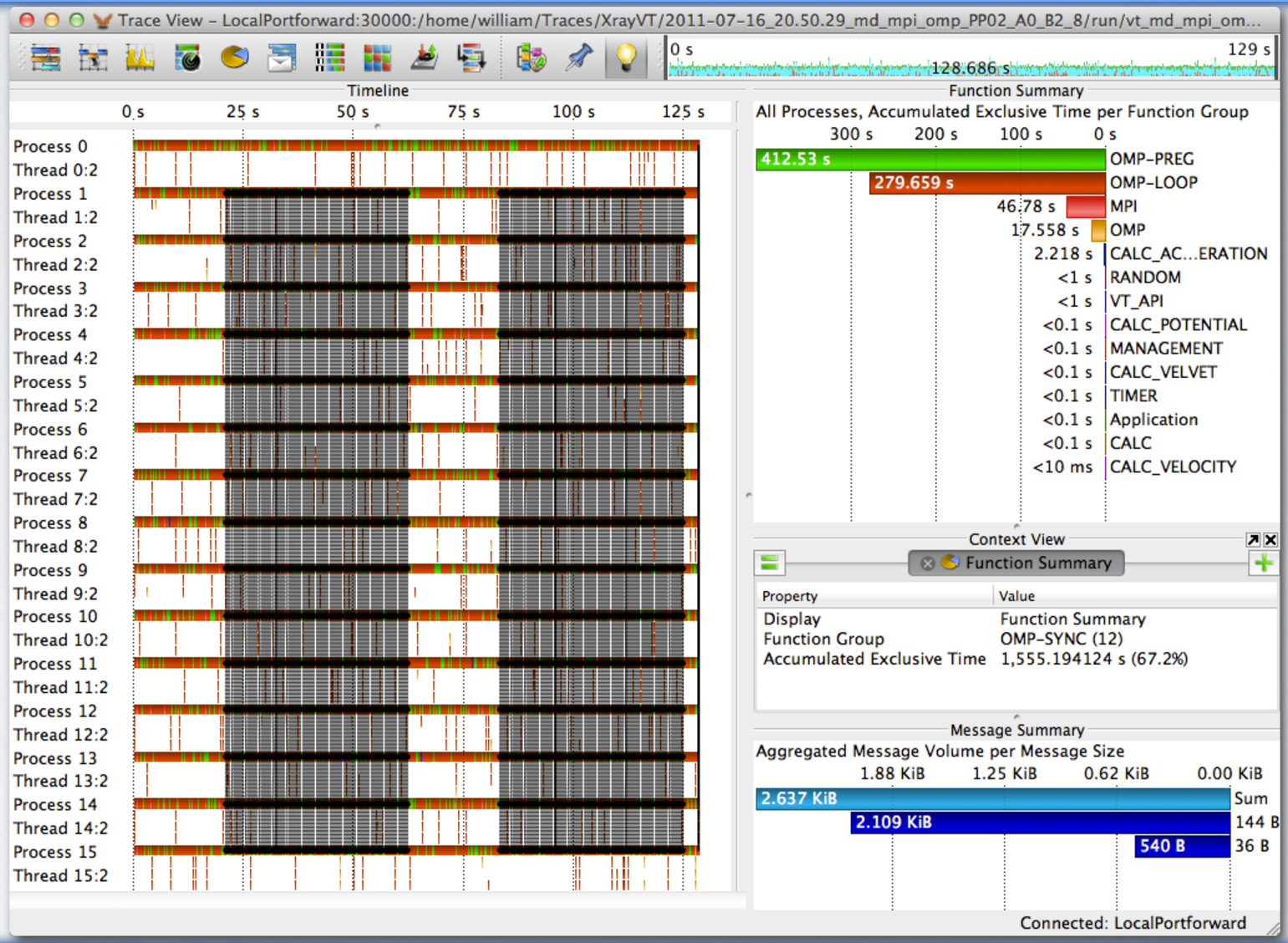
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

PARALLEL ANALYSIS WITH VAMPIR



TECHNISCHE UNIVERSITÄT DRESDEN

ZIH
Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

PARALLEL ANALYSIS WITH VAMPIR

```
allocate(fj(3,0:n-1))
do 100 i=myrank,n-2,nprocs
   fi(:)=0.0d0
   !$omp parallel do private(r2,k,xx,r,fc),\
         reduction(+:fi), schedule(runtime)
   do 90 j=i+1,n-1
      !--- A-Block---
.
.
.
.
.
.
```

```
!$omp parallel private(nthrd,nj,nr,j0,
                       j1,xx,r2,r,fc,fi)
nthrd = omp_get_num_threads()
allocate(fi(3,0:n-1))
!$omp do schedule(static,1)
do 110 ithrd=0,nthrd-1
do 100 i=myrank,n-2,nprocs
   fi(:,i)=0.0d0
   nj=(n-i-1)/nthrd
   nr=mod(n-i-1,nthrd)
   j0=(i+1)+ithrd*nj+min(ithrd,nr)
   j1=(i+1)+(ithrd+1)*nj+min(ithrd+1,nr)-1
   do 90 j=j0,j1
```


TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

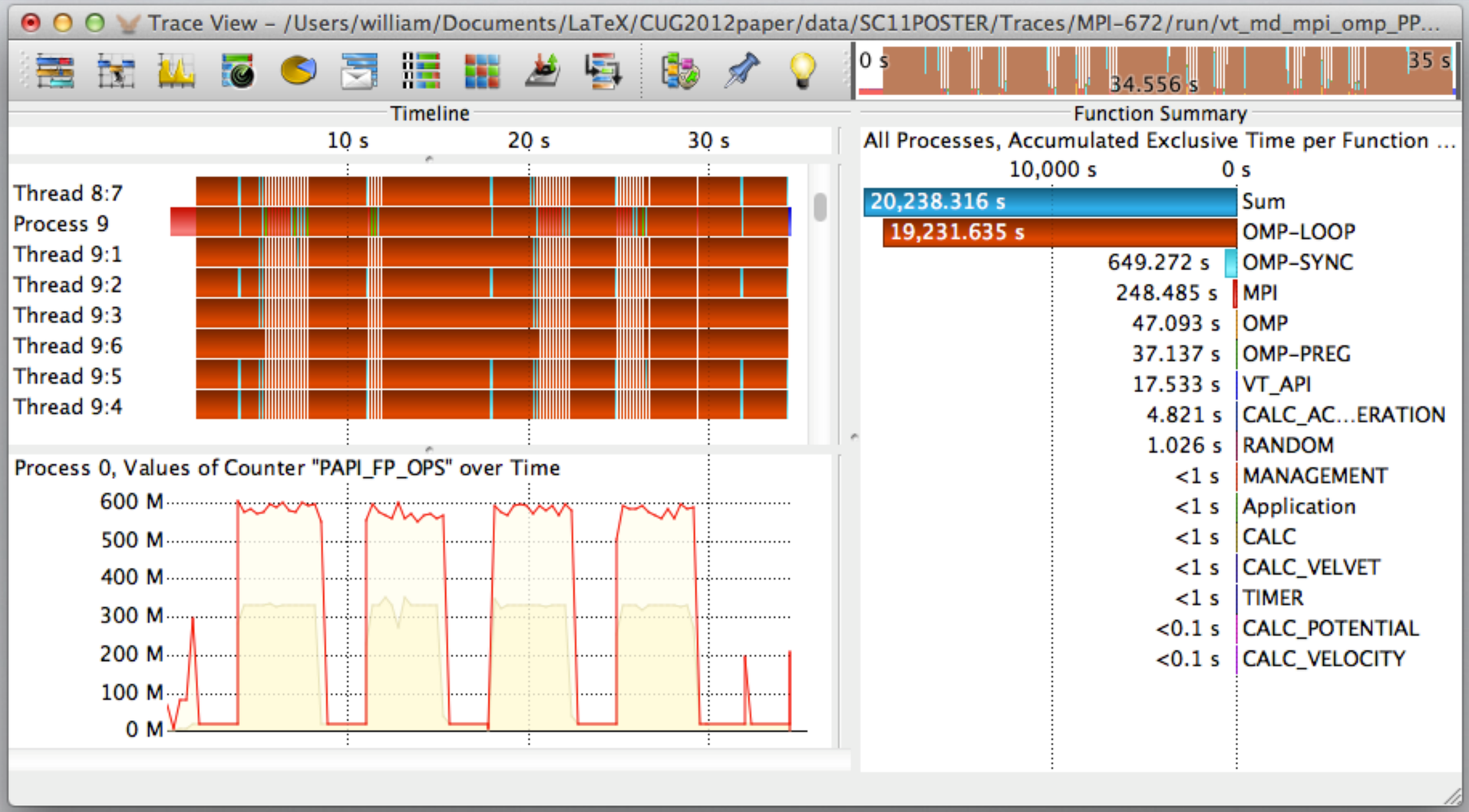
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

PARALLEL ANALYSIS WITH VAMPIR



TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

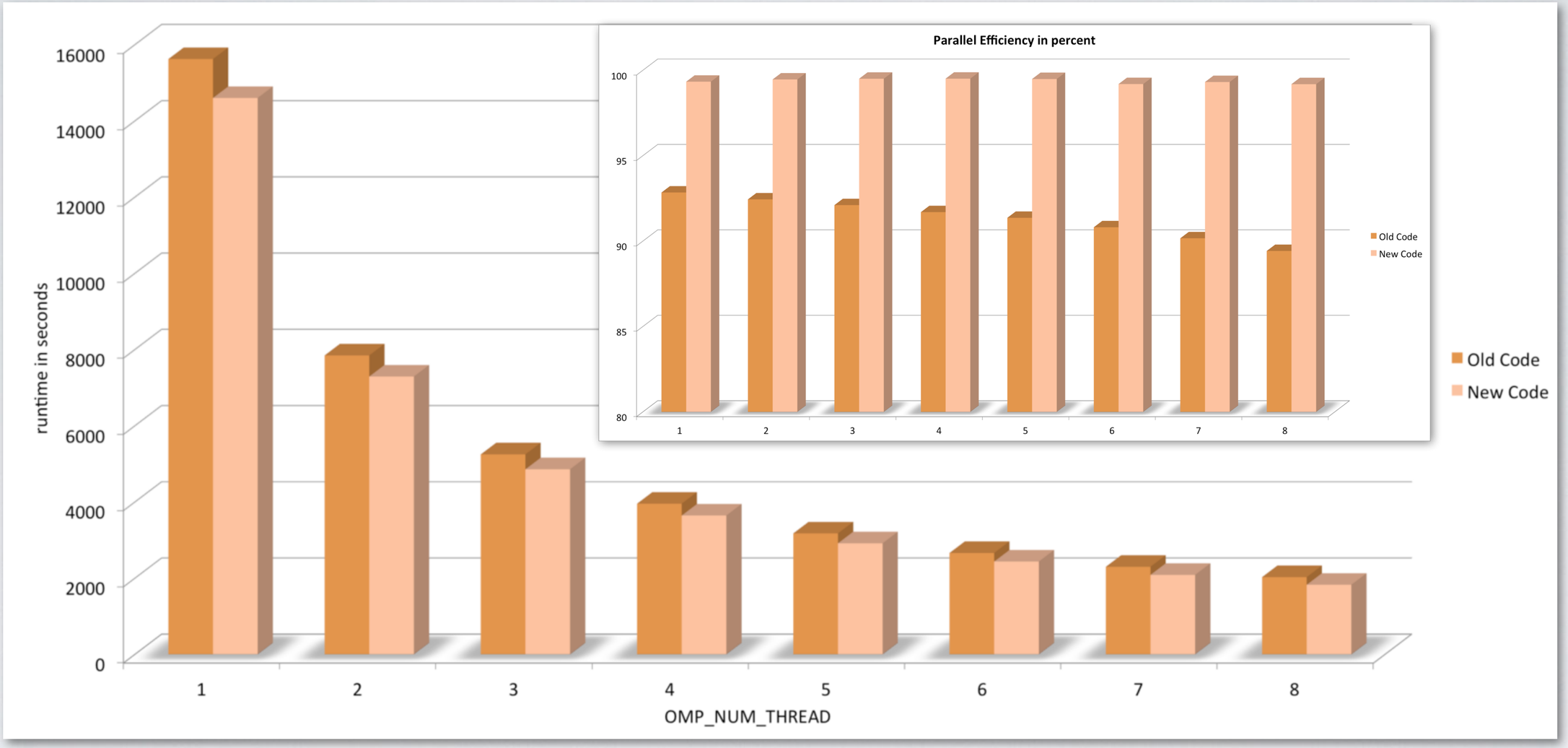
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

OPENMP CODE OPTIMIZATION



Comparison of OpenMP Versions using 1 Node and 1-8 Cores

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH
Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

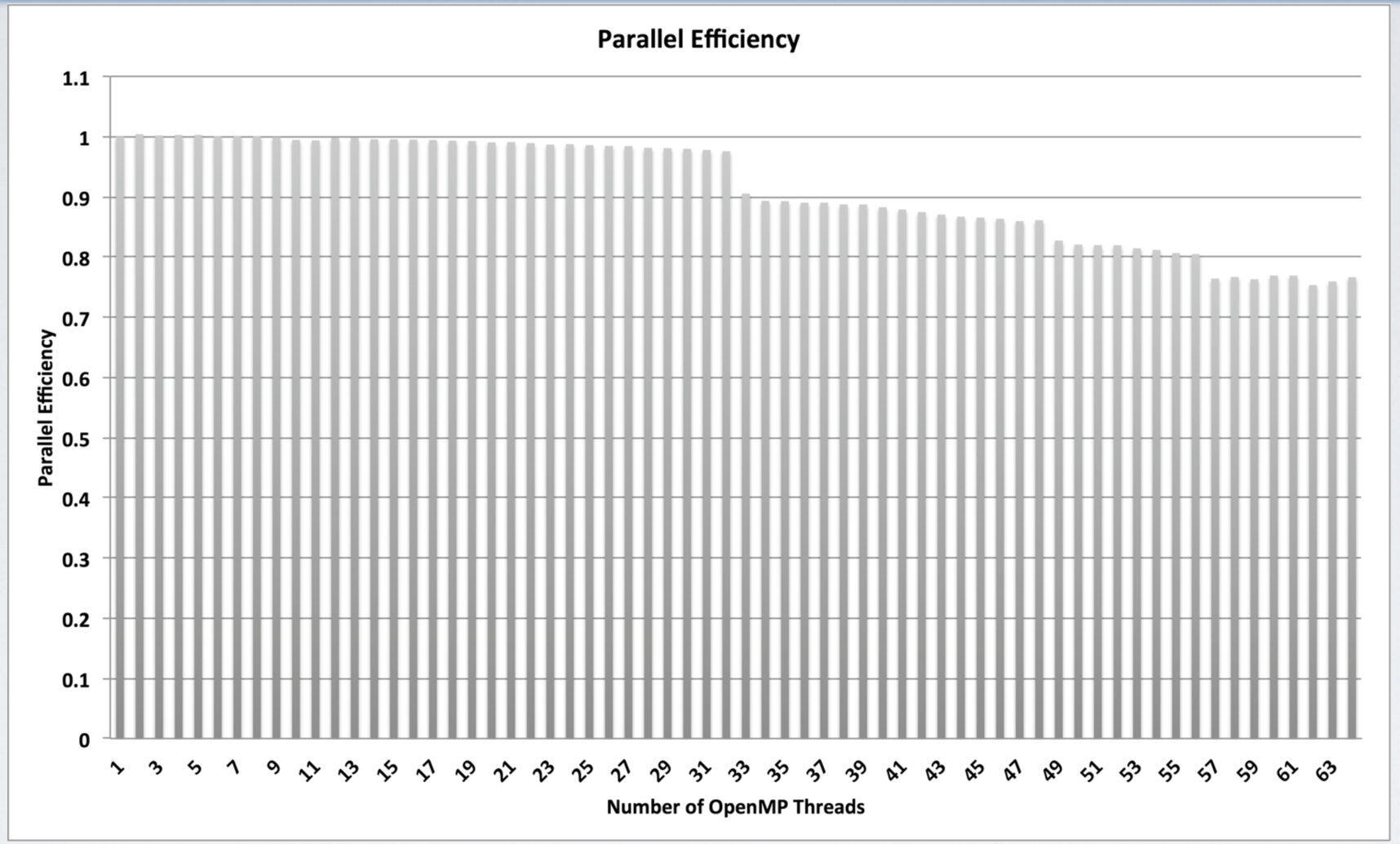
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

OPENMP CODE OPTIMIZATION



AMD Interlagos (Cray XE6, BigRed II, Indiana University)

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

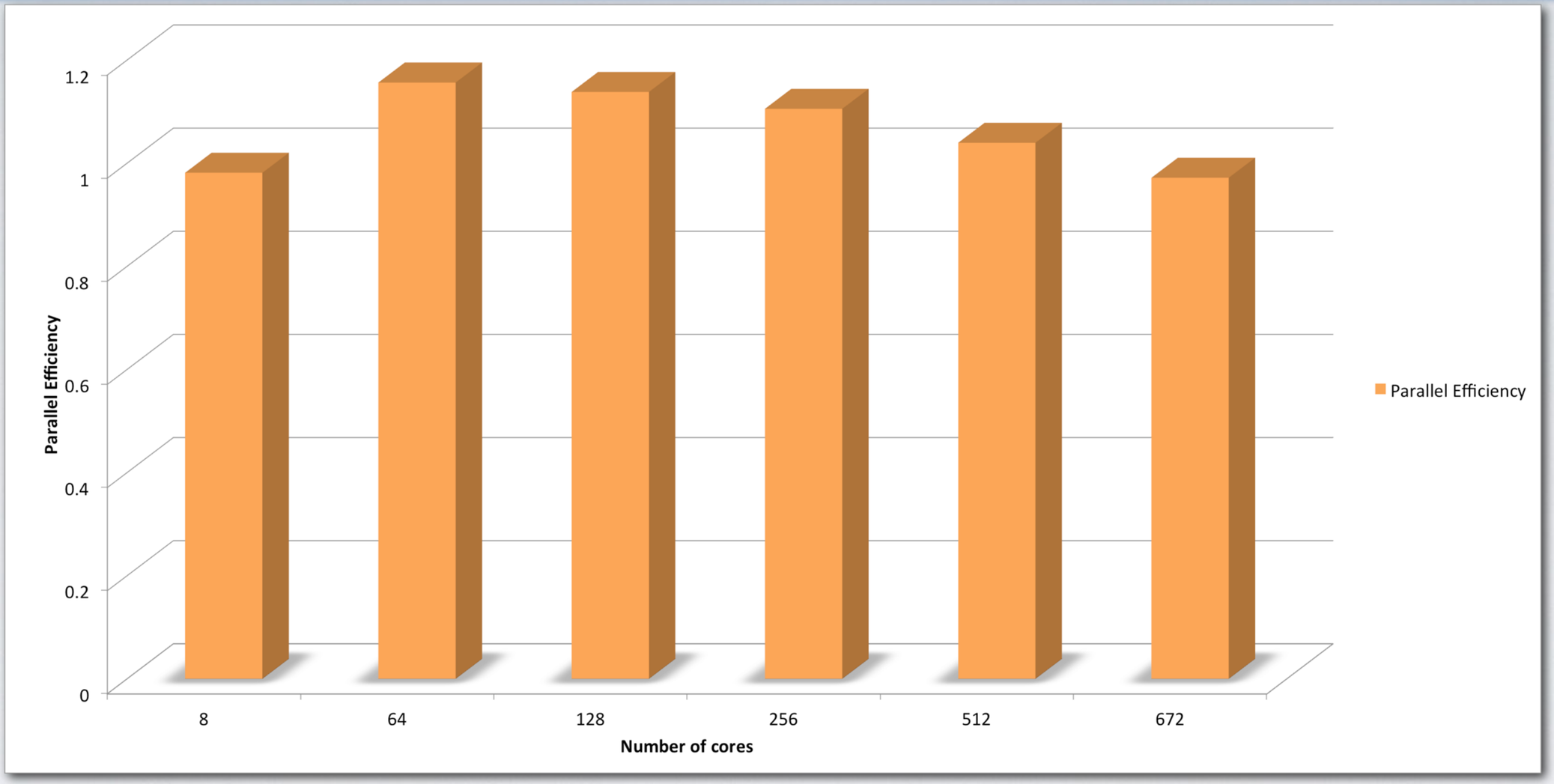
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SCALABILITY STUDIES ON XRAY



Parallel Efficiency (Hybrid runs)

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH
Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

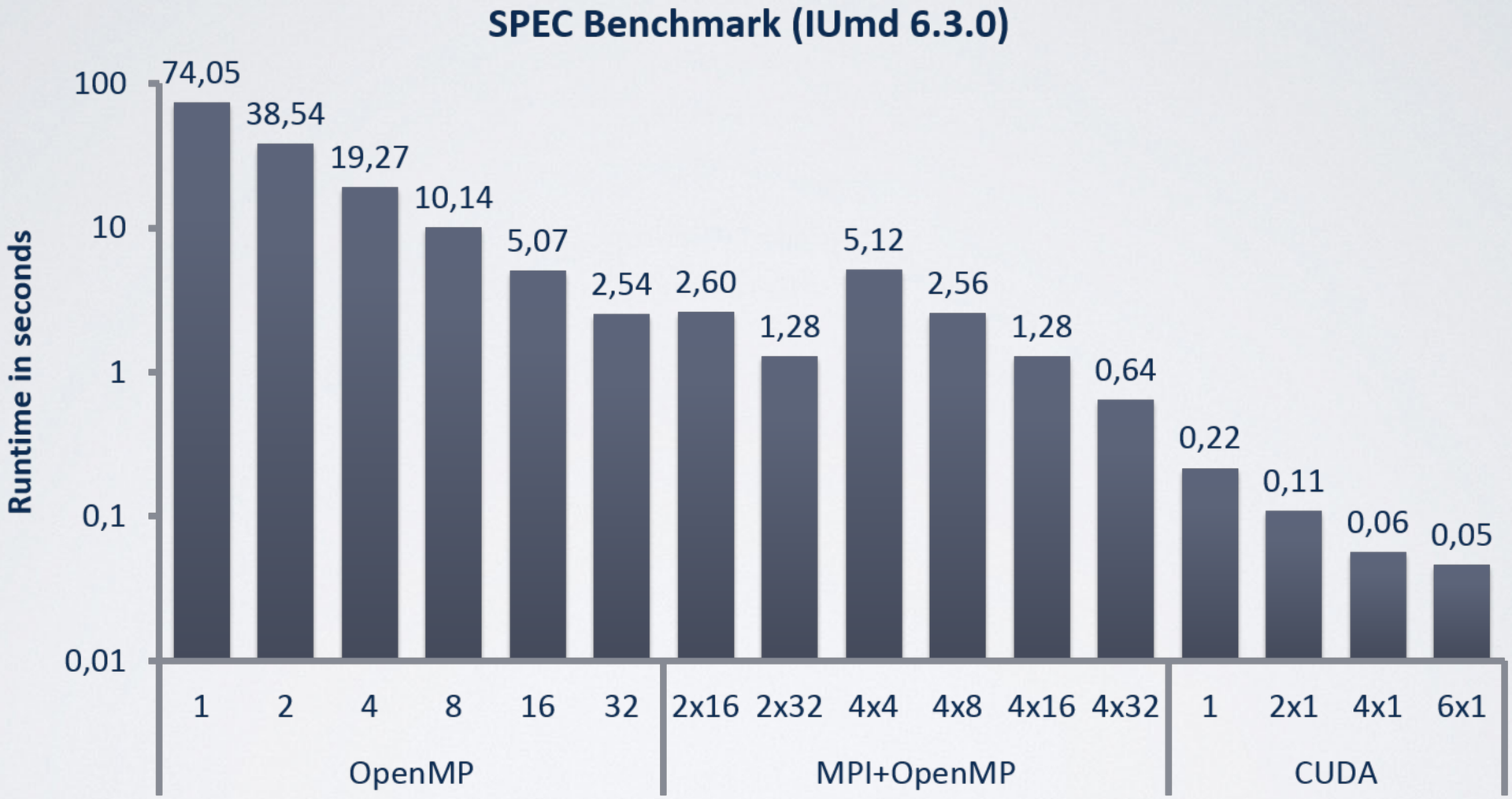
IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY

SCALABILITY STUDIES ON XRAY



TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Technische Universität Dresden
Zentrum für Informationsdienste
und Hochleistungsrechnen (ZIH)

01062 Dresden, Germany

E-mail: contact@vampir.eu
Web: http://www.vampir.eu/

VAMPIR

IU Bloomington
Pervasive Technology Institute

2719 East 10th Street
Bloomington, Indiana 47408

E-mail: pti@indiana.edu
Web: http://pti.iu.edu/

INDIANA UNIVERSITY