Adaptation dynamique des optimisations : versions interchangeables

Lénaïc Bagnères, Cédric Bastoul

Laboratoire de Recherche en Informatique (LRI)
Inria Grand Large
Université Paris-Sud 11

Laboratoire ICube - ICPS
Inria CAMUS
Université de Strasbourg

7e rencontres de la communauté française de compilation
6 décembre 2013

Introduction

- Once upon a time, life was easy for parallel programs...
  - One program running, one architecture, one context, one user

- In the multicore era, welcome to the jungle!
  - A large set of target architectures is possible
  - High dataset variations (user and/or architecture dependent)
  - Several parallel programs are executing at the same time

Execution Context & Related Work

Program Must Adapt (Transparently) to the Execution Context

- Target architecture

  (e.g. number of cores, size of caches, accelerator type)
- Dataset properties

  (e.g. dataset size)
- System load

  (e.g. number of running process, processor load, memory access)

Related Work

- Choice based on compile time profiling [Pradelle et al., 2011]

  Possible prediction error, switching not possible during kernel execution
- `LeTSeE` [Pouchet et al., 2008]

  Choses the best version for a given architecture and context
- One application with `VMAD` [Jimborean et al., 2012]

  Select the best version at runtime, hight overhead, user develops and chooses versions
- `EvolveTile` [Tavarageri et al., 2011]

  Perform an dynamic tile size selection
- Number of threads depends on the external workload [Emani et al., 2013]

Our Approach: Switchable Scheduling

In a nutshell:

- Prepare a pertinent set of versions
- Versions have meeting points were it is possible to switch from one to another
- A runtime analysis drives to the "best" version at each meeting point

Properties:

- Static / dynamic approach
- Low overhead
- Benchmarking computations contribute to the final solution:

  no backtrack

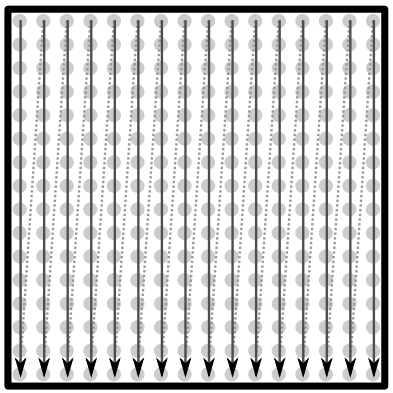
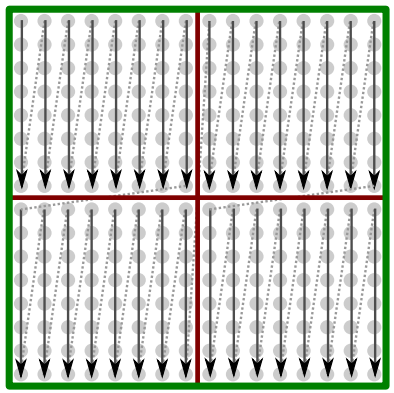
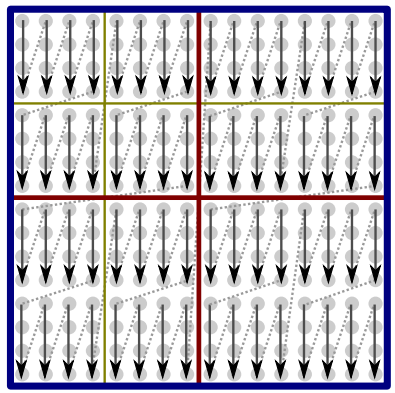
Building Switchable Scheduling

- Leveraging our expertise at generating program versions using the polyhedral model [Pouchet et al., 2007]

- Formal definition:
  - A version is defined as a set of scheduling relations
  - A meeting iteration $\vec{m}$ exists between two versions V and V' iff $\forall S, (\mathcal{T}_s \wedge \vec{x}_S^{out} \preceq \vec{m})_{in} - (\mathcal{T}'_s \wedge \vec{x}_S^{out} \preceq \vec{m})_{in} = \varnothing$

- Pratical building

  First dimensions of the scheduling are either the same, or tiled

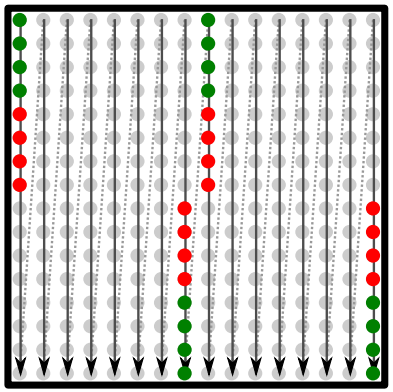
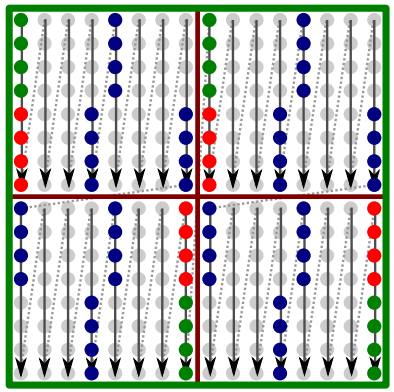
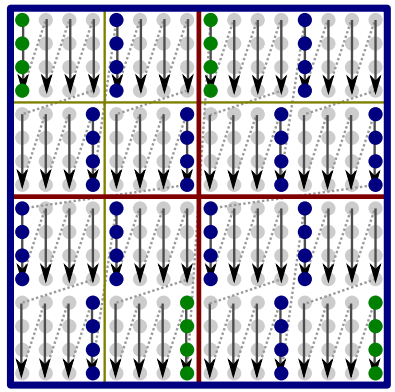
Building Switchable Scheduling

- Leveraging our expertise at generating program versions using the polyhedral model [Pouchet et al., 2007]

- Formal definition:
  - A version is defined as a set of scheduling relations
  - A meeting iteration $\vec{m}$ exists between two versions V and V' iff $\forall S, (\mathcal{T}_s \wedge \vec{x}_S^{out} \preceq \vec{m})_{in} - (\mathcal{T}'_s \wedge \vec{x}_S^{out} \preceq \vec{m})_{in} = \varnothing$

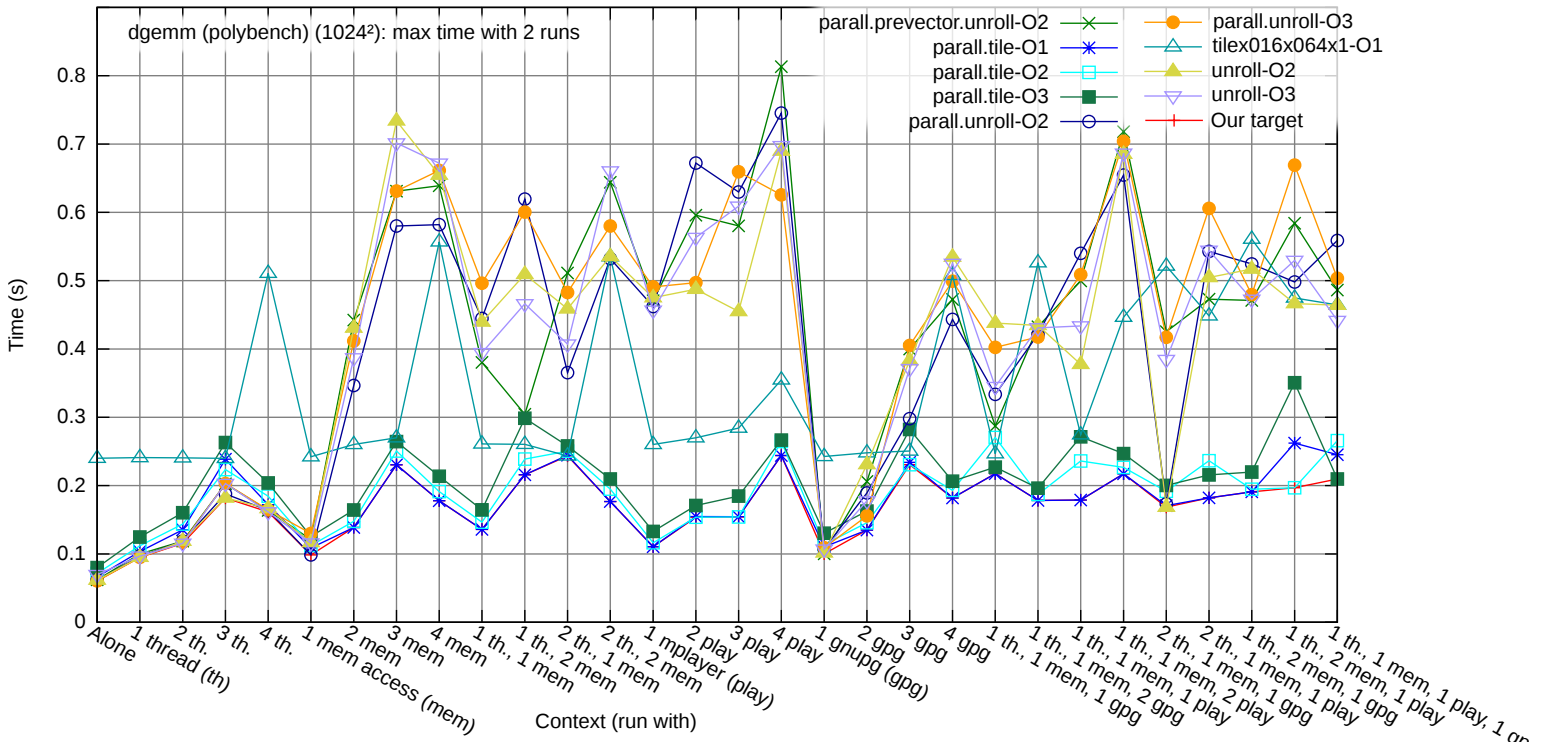
- Pratical building

  First dimensions of the scheduling are either the same, or tiled

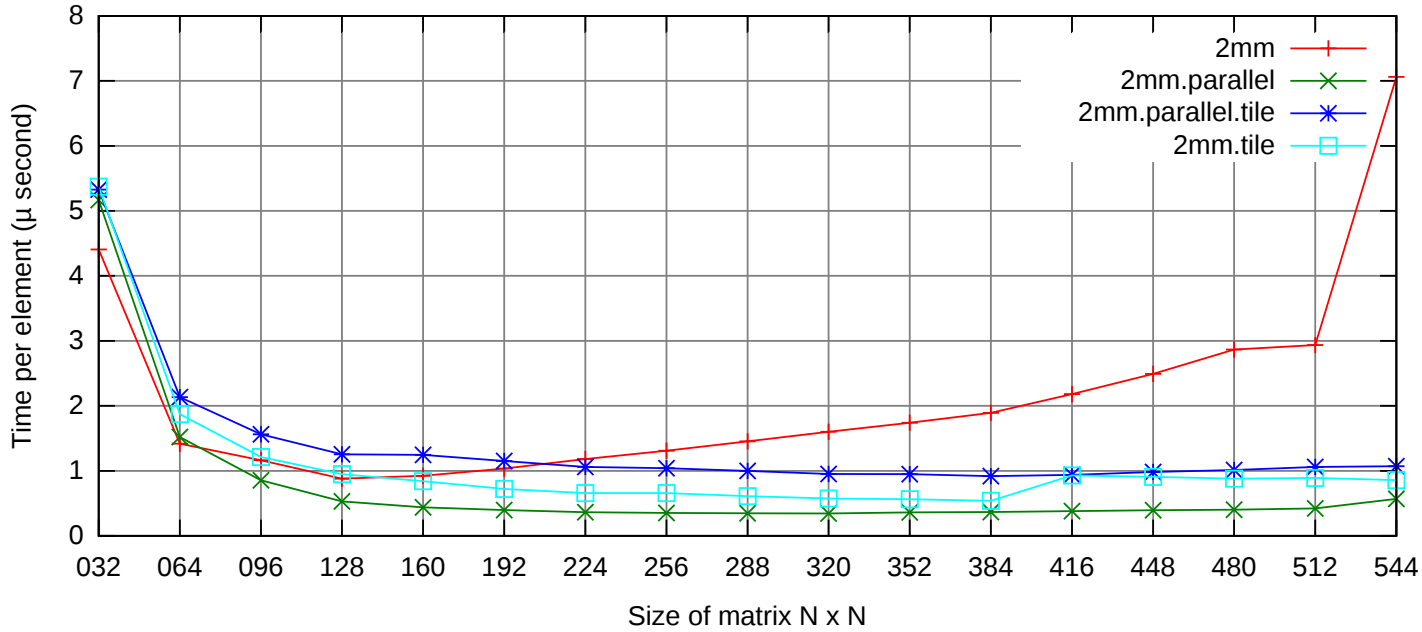
Building Switchable Scheduling: Static Part

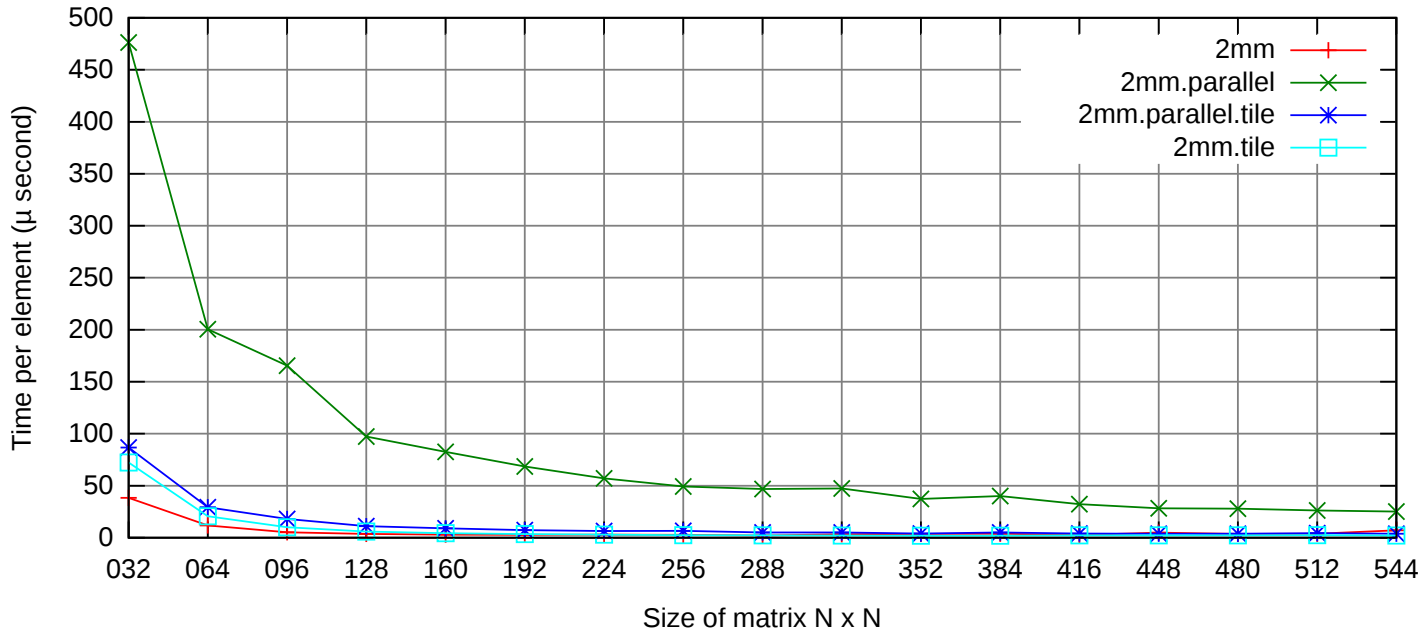
- Pre-selection of versions:
  - Switchable versions building engine based on `PoCC` and `Pluto`
  - Generate versions: sequential, parallel, tiled, etc.
- Pre-selected pertinents contexts
  - Architecture (number of cores, size of caches)
  - Aggressive processor use or/and memory access processes
- Selecting pertinents versions:
  - Execution on artificial contexts
  - Selection of a restricted set of complementary "well-behaving" versions



On Intel Quad Core Q6600 2.4GHz, Linux 3.2, GCC 4.7.2

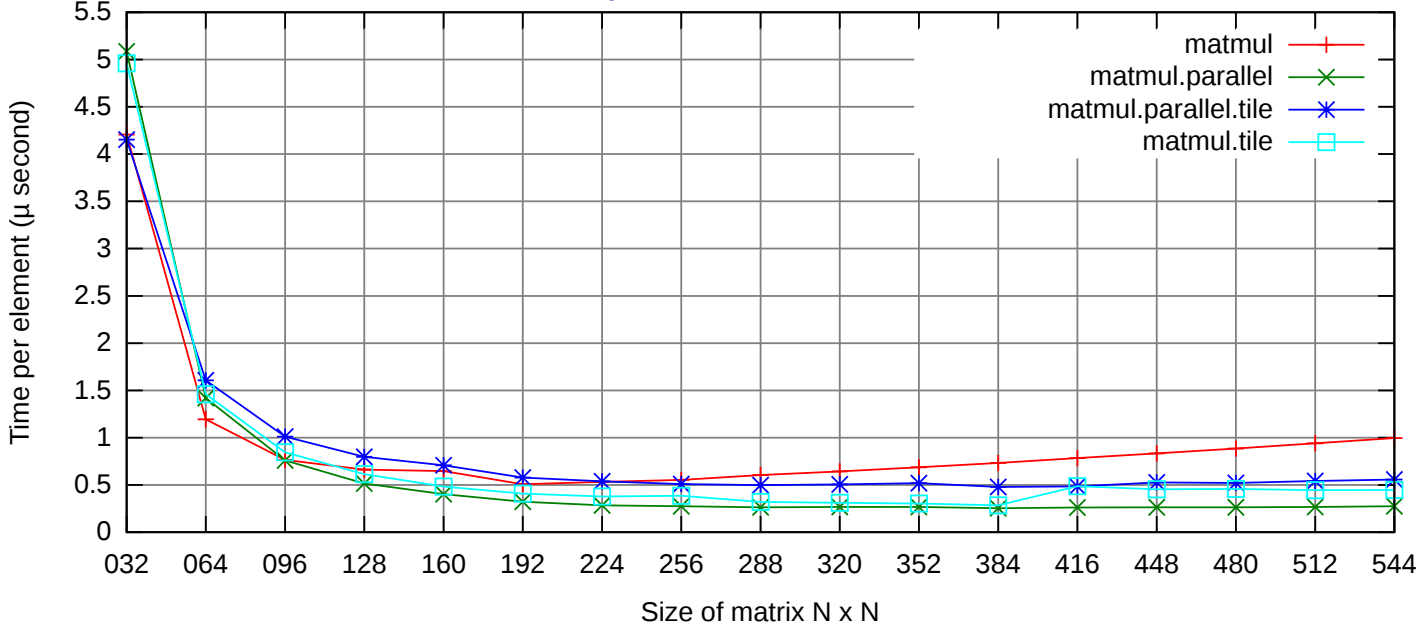
System load context: 2mm without and with strong system load on Xeon[1]

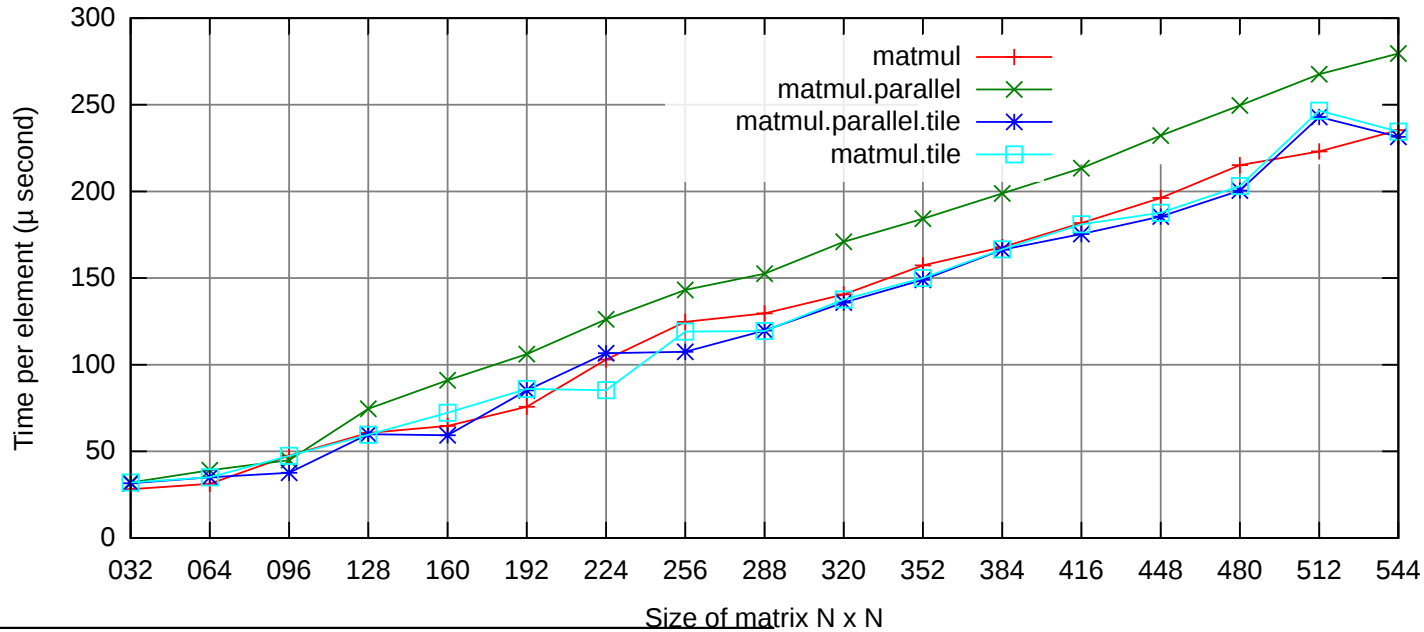




[1] Xeon: Two Intel Xeon CPU E5645 6 cores 2.40GHz, Linux 3.2, GCC 4.6.3

Architecture context: Multiplication matrix on Xeon[2] and ARMv6[3]





[2]Xeon: Two Intel Xeon CPU E5645 6 cores 2.40GHz, Linux 3.2, GCC 4.6.3

[3]ARMv6: LG GW620, ARMv6 1 core 600Mhz, Linux 2.6.32, GCC ARM Linux GNU ABI 4.7.2

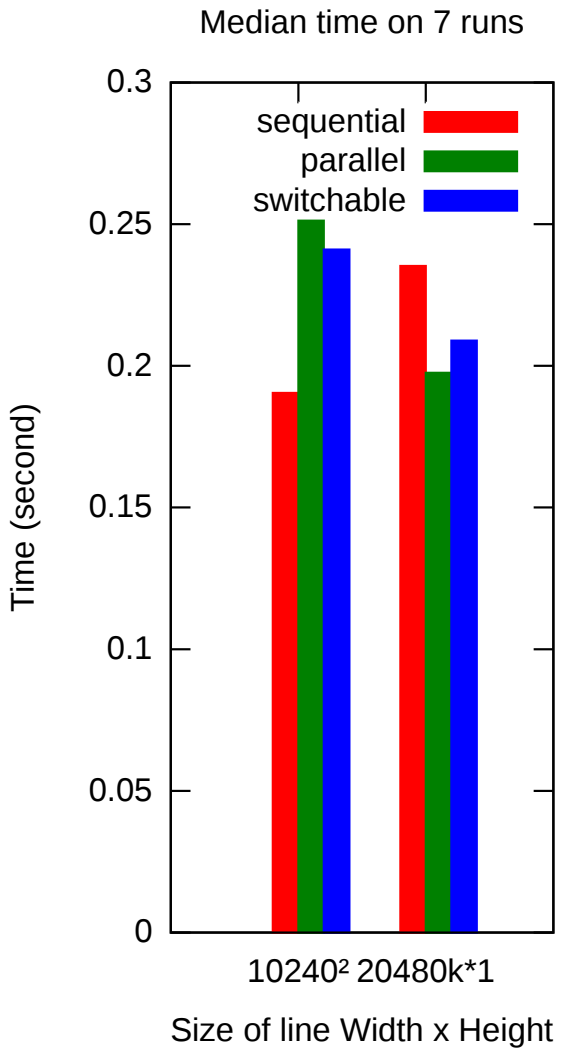
Building Switchable Scheduling: Dynamic Part

(a) selecting a version and switching at runtime

```
while (i < N)
// Sampling
for (v = 0 → versions.size())
    t0 = now()
    // Execute one version
    versions[v](i, i + sample_size)
    t1 = now()
    times[v] = t1 - t0

// Computing
// Execute previous best version
version[min(times)](i, i + compute_size)
```

(b) actual performance on line bench



On Intel Core 2 Quad Q9000 2.00GHz, Linux 3.2, GCC 4.7.2

Conclusion

- Lessons learned so far:
  - Already useful to converge to the best version with respect to the context
  - Low overhead techniques demonstrated essential (but we are not good enough...)
  - Too often, we still get one version to rule them all (no switching after convergence)

- Ongoing work and todo list:
  - More aggressive selection of versions
  - Refinement of switchable version set study at compile time using collective tuning ([Fursin, 2009], `http://ctuning.org`)
  - Refinement of switchable version set at runtime using dynamic tests à la [Pradelle et al., 2011]
  - Background pre-sampling

References

Emani, M., Wang, Z. and O'Boyle, M. (2013).

Smart, adaptive mapping of parallelism in the presence of external workload.
In Code Generation and Optimization (CGO), 2013 IEEE/ACM International Symposium on pp. 1–10,.

Fursin, G. (2009).

Collective Tuning Initiative: automating and accelerating development and optimization of computing systems.
In Proceedings of the GCC Developers' Summit.

Jimborean, A., Mastrangelo, L., Loechner, V. and Clauss, P. (2012).

VMAD: an Advanced Dynamic Program Analysis & Instrumentation Framework.
In CC - 21st International Conference on Compiler Construction, (O'Boyle, M., ed.), vol. 7210, of Lecture Notes in Computer Science pp. 220–237, Springer, Tallinn, Estonia.

Pouchet, L.-N., Bastoul, C., Cohen, A. and Cavazos, J. (2008).

Iterative optimization in the polyhedral model: Part II, multidimensional time.
In ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'08) pp. 90–100, ACM Press, Tucson, Arizona.

Pouchet, L.-N., Bastoul, C., Cohen, A. and Vasilache, N. (2007).

Iterative optimization in the polyhedral model: Part I, one-dimensional time.
In IEEE/ACM Fifth International Symposium on Code Generation and Optimization (CGO'07) pp. 144–156, IEEE Computer Society press, San Jose, California.

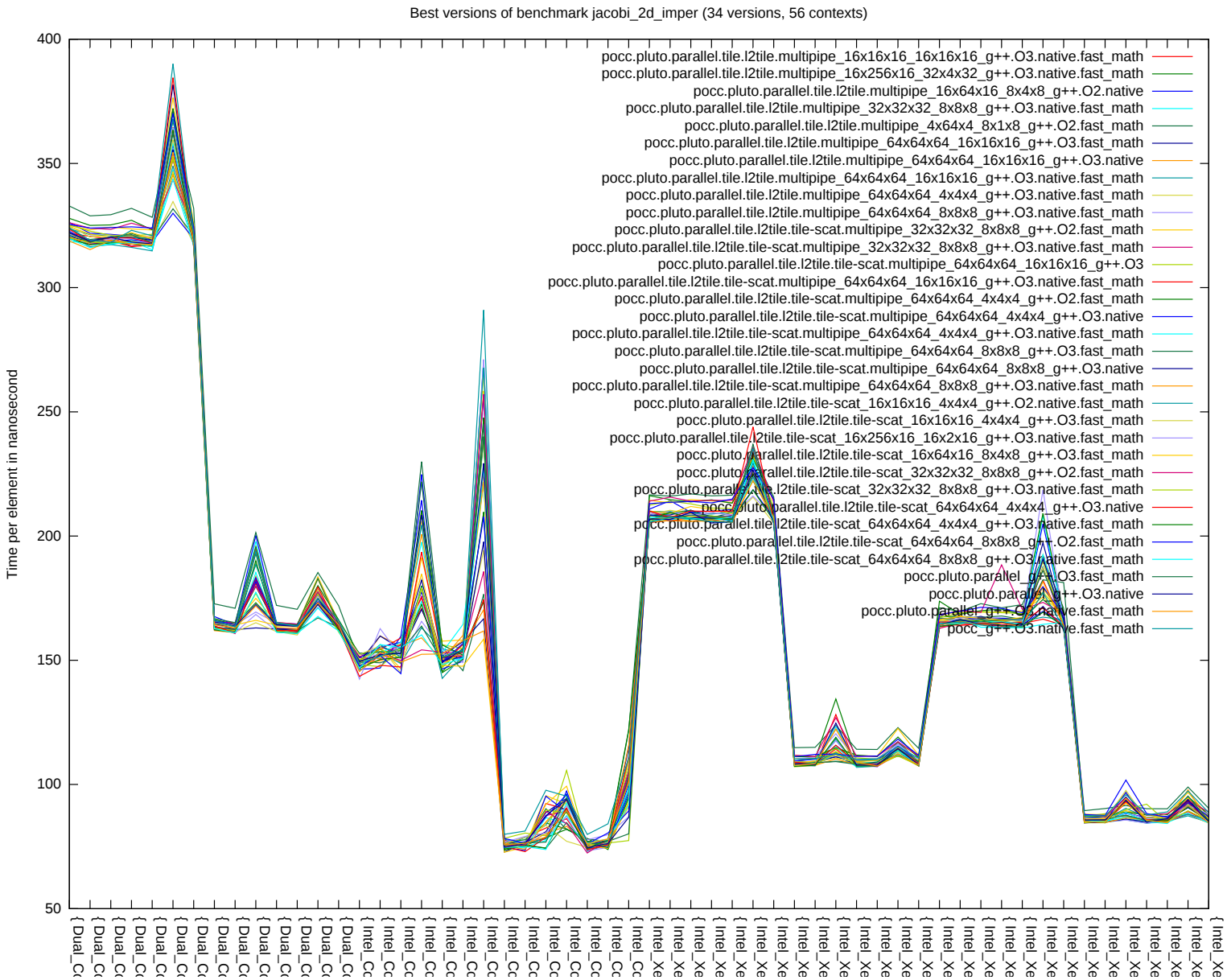
Pradelle, B., Clauss, P. and Loechner, V. (2011).

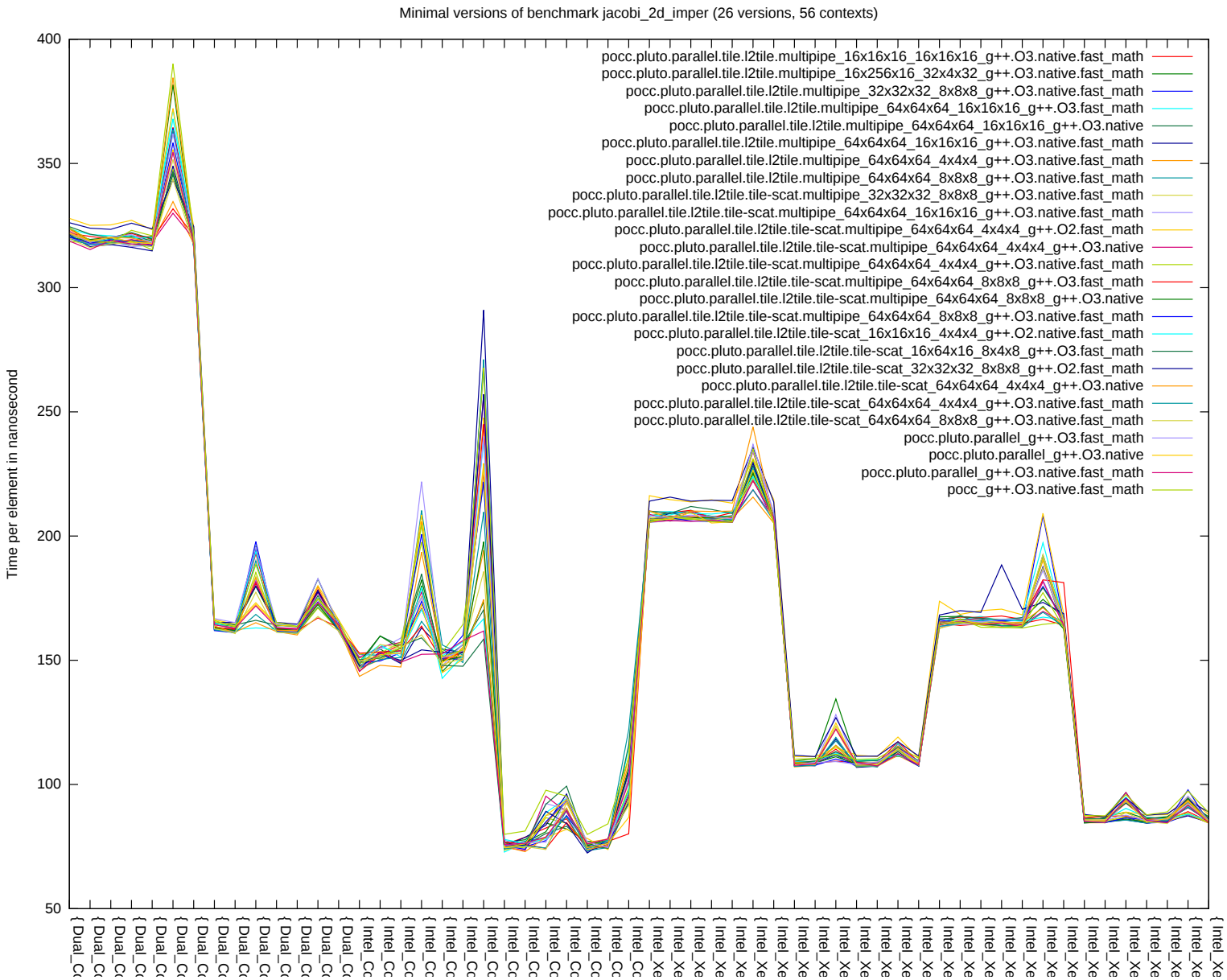
Adaptive Runtime Selection of Parallel Schedules in the Polytope Model.
In 19th High Performance Computing Symposium - HPC 2011 ACM/SIGSIM, Boston, United States.

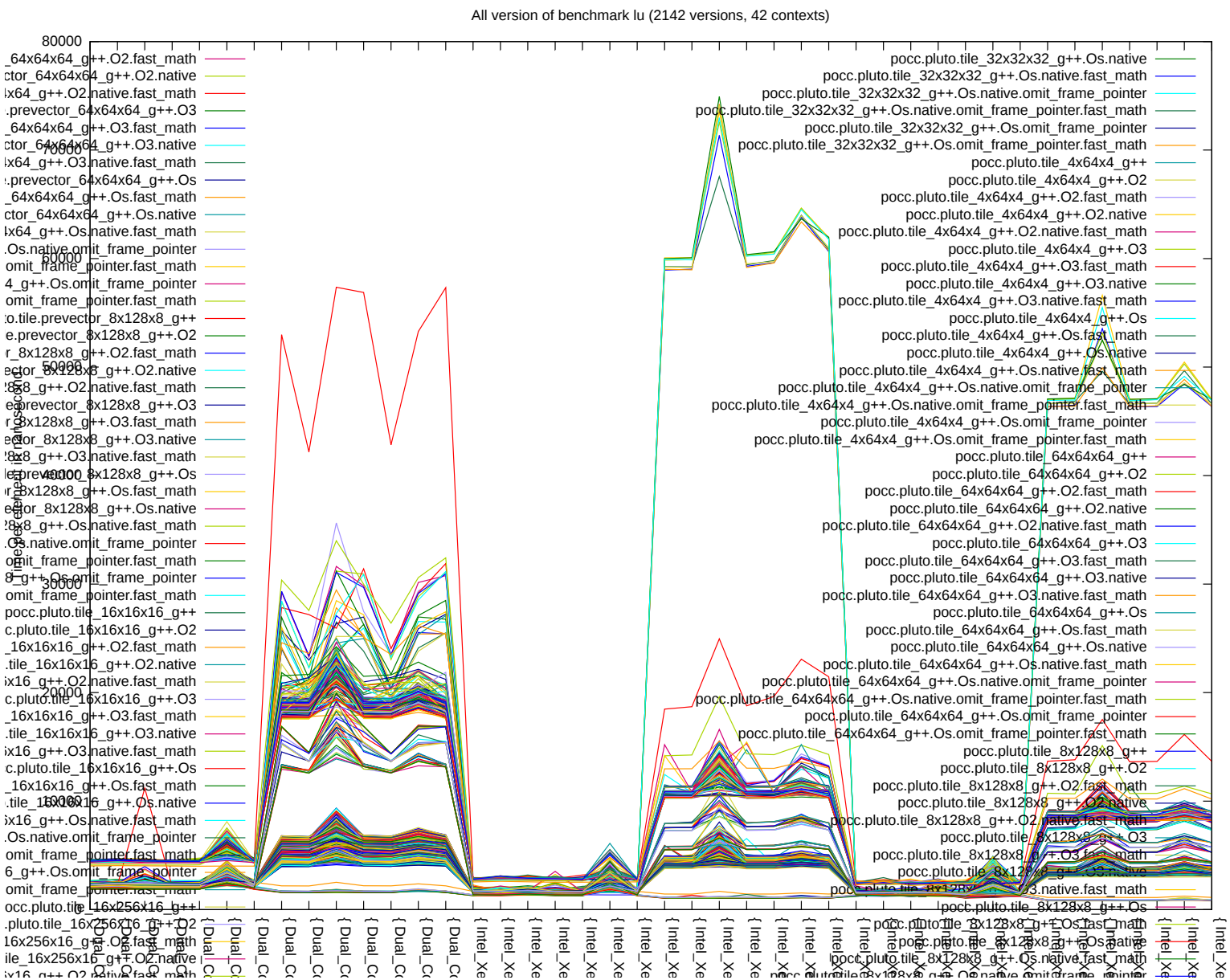
Tavarageri, S., Pouchet, L.-N., Ramanujam, J., Rountev, A. and Sadayappan, P. (2011).

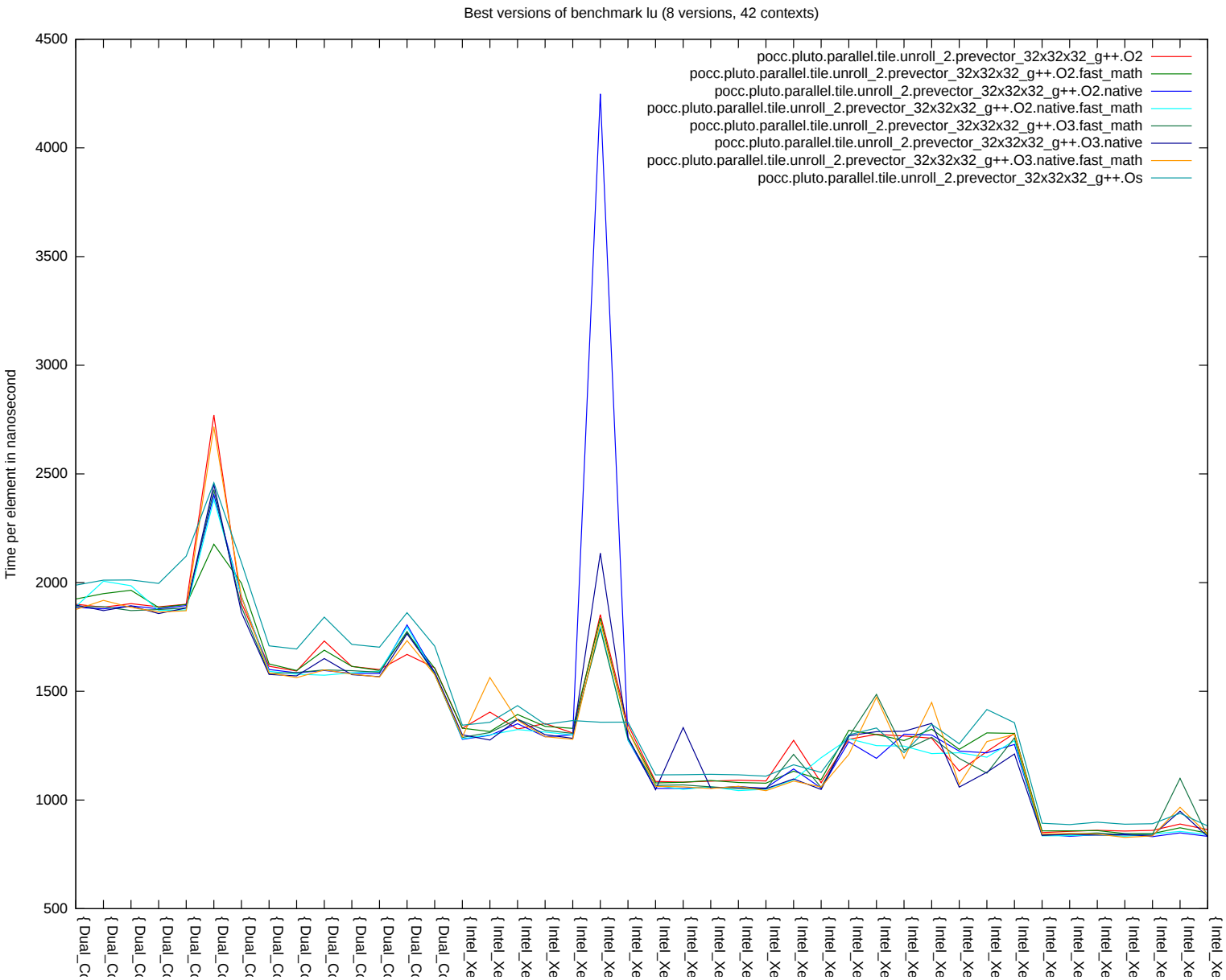
Dynamic Selection of Tile Sizes.
In 18th annual IEEE International Conference on High Performance Computing (HiPC'11) IEEE Computer Society Press, Bangalore, India.

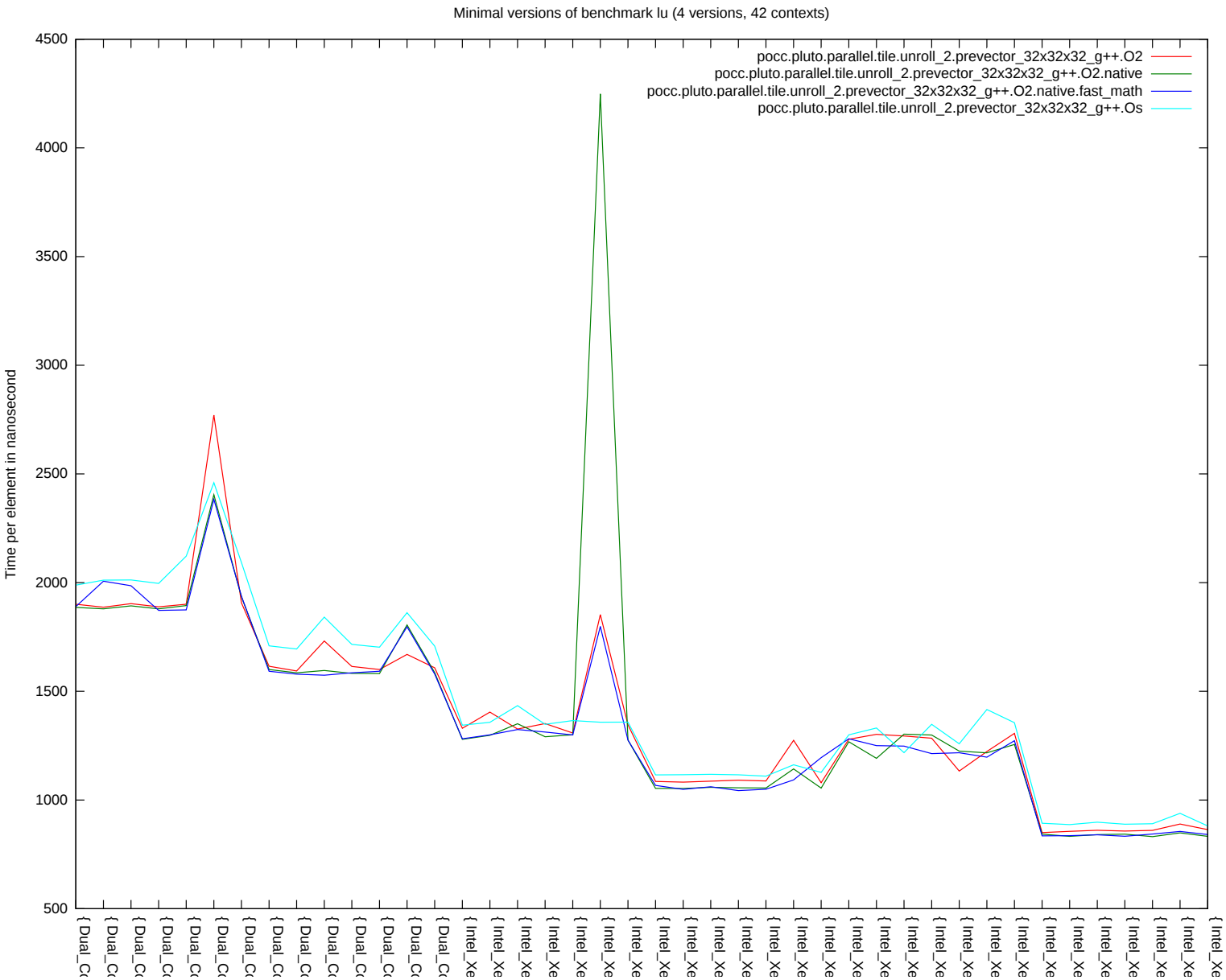

All version of benchmark jacobi_2d_imper (867 versions, 56 contexts)
1000
900
800
700
600
500
400
300
200
100
0
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O2
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O2.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O2.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O2
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O2.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O2.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O2
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O2.native
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O2.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O3
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.omit_frame_pointer
pocd.pluto.parallel.tile.l2tile.tile-scat_8x128x8_16x2x16_g++.Os.omit_frame_pointer.fast_math
Dual Core Penryn
Intel Xeon

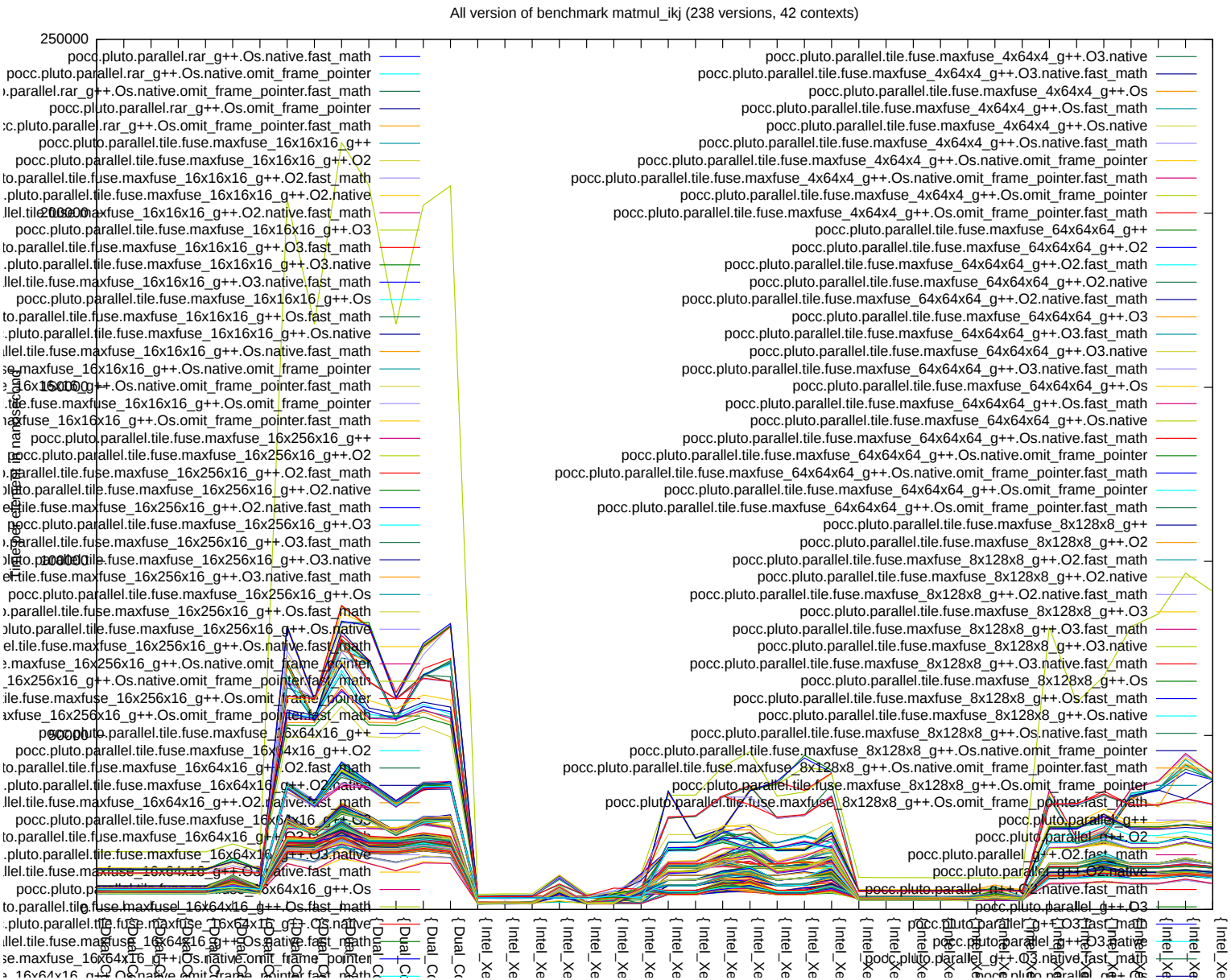

Best versions of benchmark jacobi_2d_imper (34 versions, 56 contexts)
Time per element in nanosecond
400
350
300
250
200
150
100
50
pocc.pluto.parallel.tile.l2tile.multipipe_16x16x16_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_16x256x16_32x4x32_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_16x64x16_8x4x8_g++.O2.native
pocc.pluto.parallel.tile.l2tile.multipipe_32x32x32_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_4x64x4_8x1x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.native
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_32x32x32_8x8x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_32x32x32_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_16x16x16_g++.O3
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x16x16_4x4x4_g++.O2.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x16x16_4x4x4_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x256x16_16x2x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x64x16_8x4x8_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_32x32x32_8x8x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_32x32x32_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel_g++.O3.fast_math
pocc.pluto.parallel_g++.O3.native
pocc.pluto.parallel_g++.O3.native.fast_math
pocc_g++.O3.native.fast_math

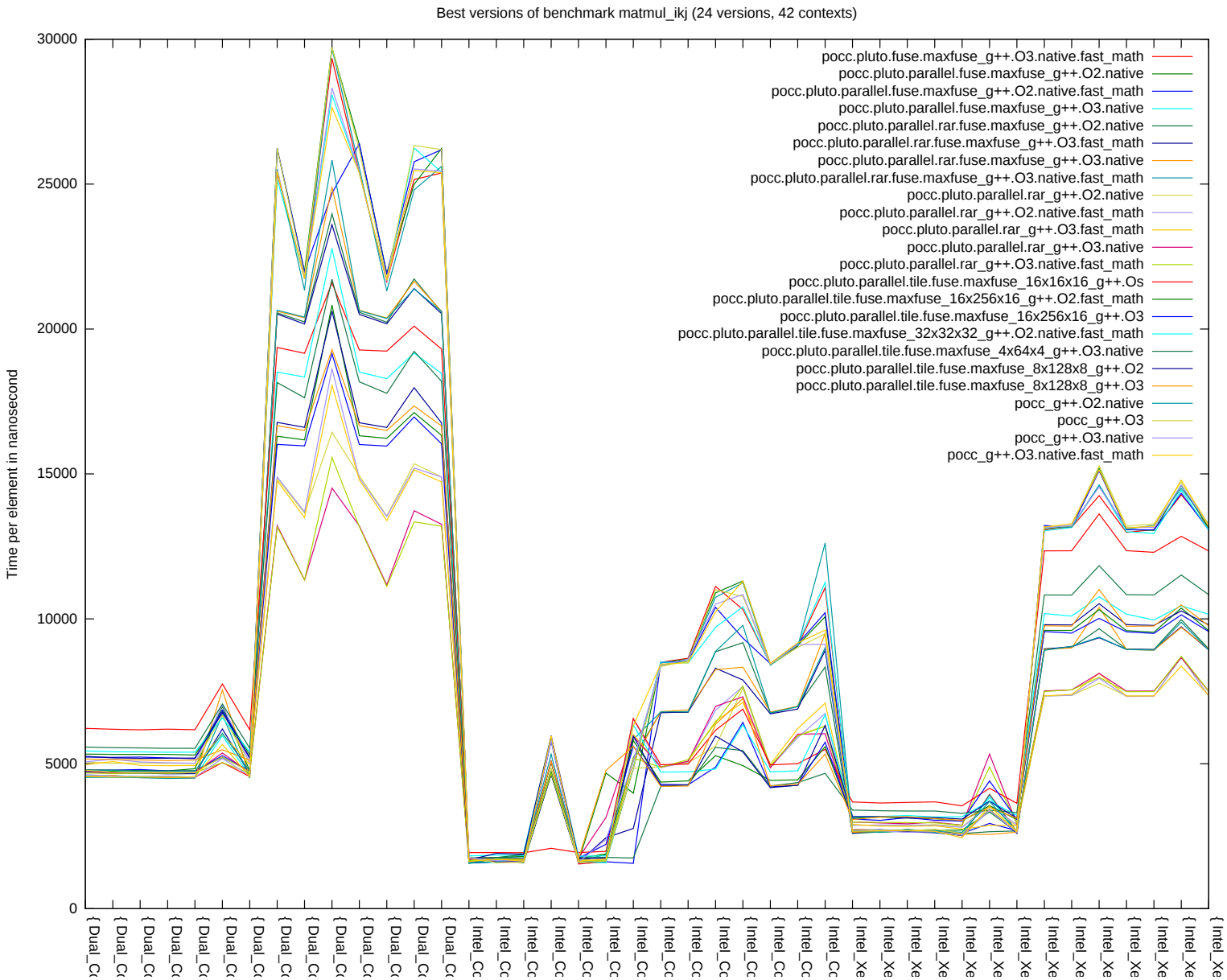

Minimal versions of benchmark jacobi_2d_imper (26 versions, 56 contexts)
Time per element in nanosecond
400
350
300
250
200
150
100
50
pocc.pluto.parallel.tile.l2tile.multipipe_16x16x16_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_16x256x16_32x4x32_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_32x32x32_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.native
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.multipipe_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_32x32x32_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat.multipipe_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x16x16_4x4x4_g++.O2.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_16x64x16_8x4x8_g++.O3.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_32x32x32_8x8x8_g++.O2.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_4x4x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.l2tile.tile-scat_64x64x64_8x8x8_g++.O3.native.fast_math
pocc.pluto.parallel_g++.O3.fast_math
pocc.pluto.parallel_g++.O3.native
pocc.pluto.parallel_g++.O3.native.fast_math
pocc_g++.O3.native.fast_math

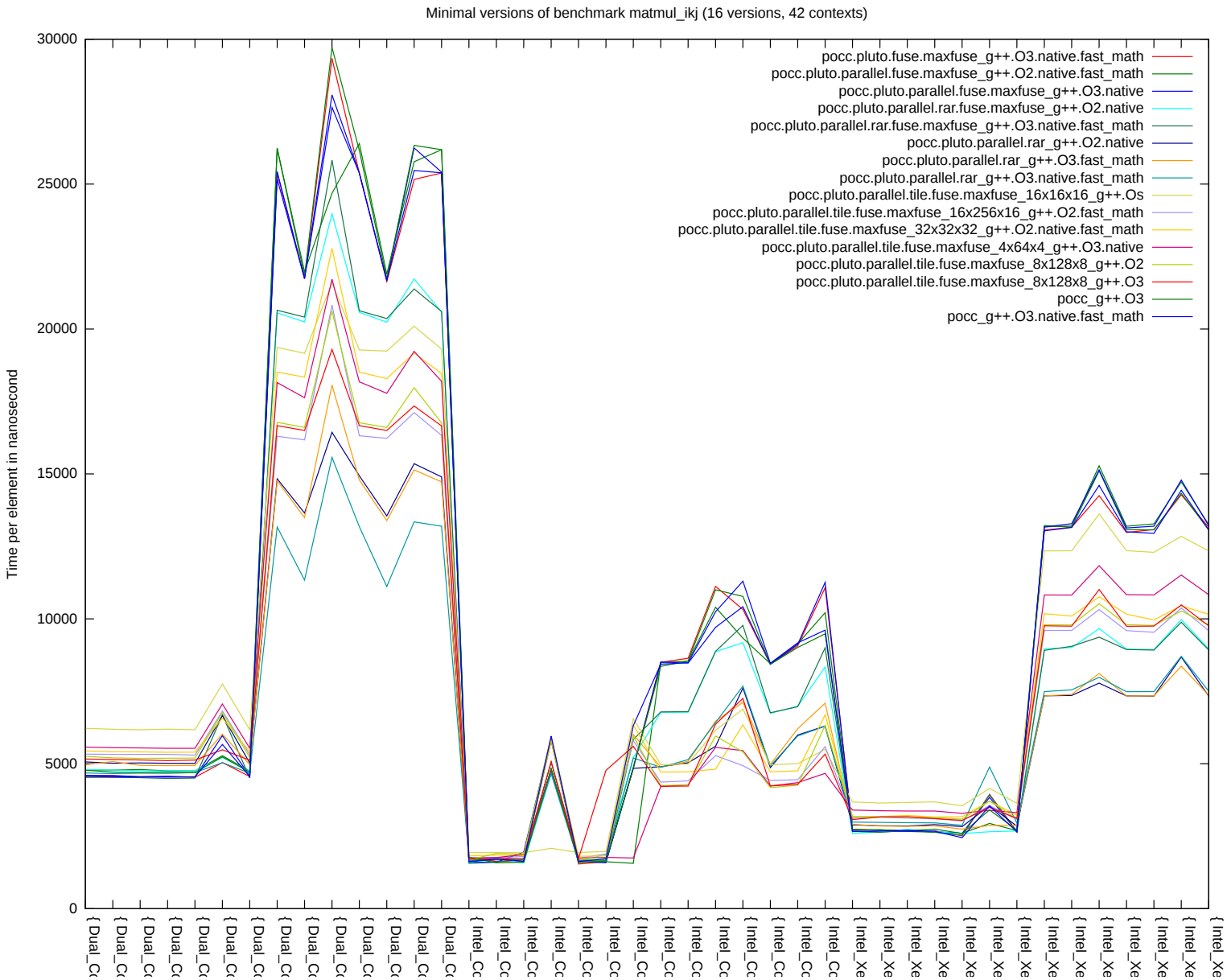

All version of benchmark lu (2142 versions, 42 contexts)
80000
70000
60000
50000
40000
30000
20000
10000
pocc.pluto.tile_32x32x32_g++.Os.native
pocc.pluto.tile_32x32x32_g++.Os.native.fast_math
pocc.pluto.tile_32x32x32_g++.Os.native.omit_frame_pointer
pocc.pluto.tile_32x32x32_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.tile_32x32x32_g++.Os.omit_frame_pointer
pocc.pluto.tile_32x32x32_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.tile_4x64x4_g++
pocc.pluto.tile_4x64x4_g++.O2
pocc.pluto.tile_4x64x4_g++.O2.fast_math
pocc.pluto.tile_4x64x4_g++.O2.native
pocc.pluto.tile_4x64x4_g++.O2.native.fast_math
pocc.pluto.tile_4x64x4_g++.O3
pocc.pluto.tile_4x64x4_g++.O3.fast_math
pocc.pluto.tile_4x64x4_g++.O3.native
pocc.pluto.tile_4x64x4_g++.O3.native.fast_math
pocc.pluto.tile_4x64x4_g++.Os
pocc.pluto.tile_4x64x4_g++.Os.fast_math
pocc.pluto.tile_4x64x4_g++.Os.native
pocc.pluto.tile_4x64x4_g++.Os.native.fast_math
pocc.pluto.tile_4x64x4_g++.Os.native.omit_frame_pointer
pocc.pluto.tile_4x64x4_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.tile_4x64x4_g++.Os.omit_frame_pointer
pocc.pluto.tile_4x64x4_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.tile_64x64x64_g++
pocc.pluto.tile_64x64x64_g++.O2
pocc.pluto.tile_64x64x64_g++.O2.fast_math
pocc.pluto.tile_64x64x64_g++.O2.native
pocc.pluto.tile_64x64x64_g++.O2.native.fast_math
pocc.pluto.tile_64x64x64_g++.O3
pocc.pluto.tile_64x64x64_g++.O3.fast_math
pocc.pluto.tile_64x64x64_g++.O3.native
pocc.pluto.tile_64x64x64_g++.O3.native.fast_math
pocc.pluto.tile_64x64x64_g++.Os
pocc.pluto.tile_64x64x64_g++.Os.fast_math
pocc.pluto.tile_64x64x64_g++.Os.native
pocc.pluto.tile_64x64x64_g++.Os.native.fast_math
pocc.pluto.tile_64x64x64_g++.Os.native.omit_frame_pointer
pocc.pluto.tile_64x64x64_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.tile_64x64x64_g++.Os.omit_frame_pointer
pocc.pluto.tile_64x64x64_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.tile_8x128x8_g++
pocc.pluto.tile_8x128x8_g++.O2
pocc.pluto.tile_8x128x8_g++.O2.fast_math
pocc.pluto.tile_8x128x8_g++.O2.native
pocc.pluto.tile_8x128x8_g++.O2.native.fast_math
pocc.pluto.tile_8x128x8_g++.O3
pocc.pluto.tile_8x128x8_g++.O3.fast_math
pocc.pluto.tile_8x128x8_g++.O3.native
pocc.pluto.tile_8x128x8_g++.O3.native.fast_math
pocc.pluto.tile_8x128x8_g++.Os
pocc.pluto.tile_8x128x8_g++.Os.fast_math
pocc.pluto.tile_8x128x8_g++.Os.native
pocc.pluto.tile_8x128x8_g++.Os.native.fast_math
Dual_C
Intel_Xe


Best versions of benchmark lu (8 versions, 42 contexts)
Time per element in nanosecond
4500
4000
3500
3000
2500
2000
1500
1000
500
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O2
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O2.fast_math
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O2.native
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O2.native.fast_math
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O3.fast_math
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O3.native
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.O3.native.fast_math
pocc.pluto.parallel.tile.unroll_2.prevector_32x32x32_g++.Os




All version of benchmark matmul_ikj (238 versions, 42 contexts)
250000
200000
150000
100000
50000
0
Time (cycles)
pocc.pluto.parallel.rar_g++.Os.native.fast_math
pocc.pluto.parallel.rar_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.rar_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.rar_g++.Os.omit_frame_pointer
pocc.pluto.parallel.rar_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O2.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O3.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O3.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O2.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_16x64x16_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O2.native
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O3.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_64x64x64_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2.native
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.native
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.native.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.native.omit_frame_pointer.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.omit_frame_pointer
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.Os.omit_frame_pointer.fast_math
pocc.pluto.parallel_g++
pocc.pluto.parallel_g++.O2
pocc.pluto.parallel_g++.O2.fast_math
pocc.pluto.parallel_g++.O2.native
pocc.pluto.parallel_g++.O2.native.fast_math
pocc.pluto.parallel_g++.O3
pocc.pluto.parallel_g++.O3.fast_math
pocc.pluto.parallel_g++.O3.native
pocc.pluto.parallel_g++.O3.native.fast_math
Dual_Core_AMD_Opteron_tm_2222_SE
Intel_Xeon_X5570


Best versions of benchmark matmul_ikj (24 versions, 42 contexts)
Time per element in nanosecond
30000
25000
20000
15000
10000
5000
0
pocc.pluto.fuse.maxfuse_g++.O3.native.fast_math
pocc.pluto.parallel.fuse.maxfuse_g++.O2.native
pocc.pluto.parallel.fuse.maxfuse_g++.O2.native.fast_math
pocc.pluto.parallel.fuse.maxfuse_g++.O3.native
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O2.native
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O3.fast_math
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O3.native
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O3.native.fast_math
pocc.pluto.parallel.rar_g++.O2.native
pocc.pluto.parallel.rar_g++.O2.native.fast_math
pocc.pluto.parallel.rar_g++.O3.fast_math
pocc.pluto.parallel.rar_g++.O3.native
pocc.pluto.parallel.rar_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O3
pocc.pluto.parallel.tile.fuse.maxfuse_32x32x32_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3
pocc_g++.O2.native
pocc_g++.O3
pocc_g++.O3.native
pocc_g++.O3.native.fast_math


Minimal versions of benchmark matmul_ikj (16 versions, 42 contexts)
Time per element in nanosecond
30000
25000
20000
15000
10000
5000
0
pocc.pluto.fuse.maxfuse_g++.O3.native.fast_math
pocc.pluto.parallel.fuse.maxfuse_g++.O2.native.fast_math
pocc.pluto.parallel.fuse.maxfuse_g++.O3.native
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O2.native
pocc.pluto.parallel.rar.fuse.maxfuse_g++.O3.native.fast_math
pocc.pluto.parallel.rar_g++.O2.native
pocc.pluto.parallel.rar_g++.O3.fast_math
pocc.pluto.parallel.rar_g++.O3.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_16x16x16_g++.Os
pocc.pluto.parallel.tile.fuse.maxfuse_16x256x16_g++.O2.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_32x32x32_g++.O2.native.fast_math
pocc.pluto.parallel.tile.fuse.maxfuse_4x64x4_g++.O3.native
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O2
pocc.pluto.parallel.tile.fuse.maxfuse_8x128x8_g++.O3
pocc_g++.O3
pocc_g++.O3.native.fast_math