Analyzing Dynamic Task-Based Applications on Hybrid Platforms: An Agile Scripting Approach

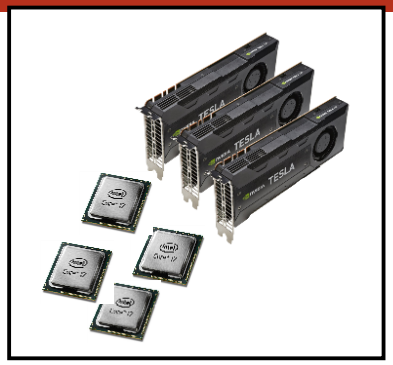
Vinícius Garcia Pinto, **Lucas Mello Schnorr**, Luka Stanisic
Arnaud Legrand, Samuel Thibault, Vincent Danjean

WSPPD Workshop
Porto Alegre, Brazil – September 4th, 2017

Context

Current HPC architectures

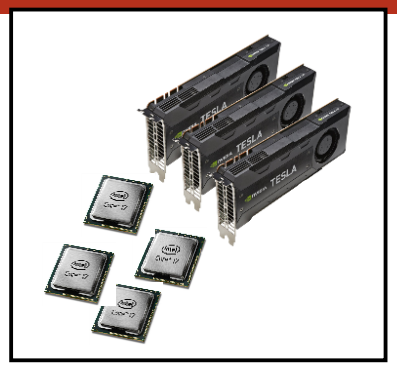
- Moving from transistors to heterogeneity scaling
- Hybrid computing resources: CPUs, GPUs, MICs



Context

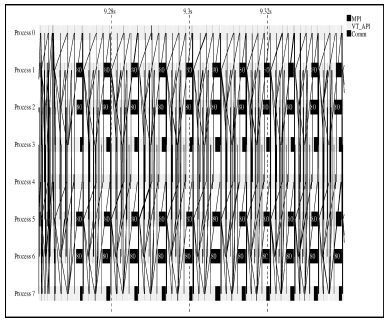
Current HPC architectures

- Moving from transistors to heterogeneity scaling
- Hybrid computing resources: CPUs, GPUs, MICs



Programming hybrid platforms

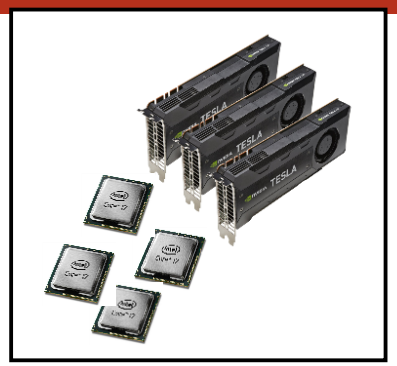
- Traditional, explicit programming models (MPI, CUDA, OpenMP, pthreads, ...)
  - ☺ Perfect control $\rightsquigarrow$ maximal achievable performance



Context

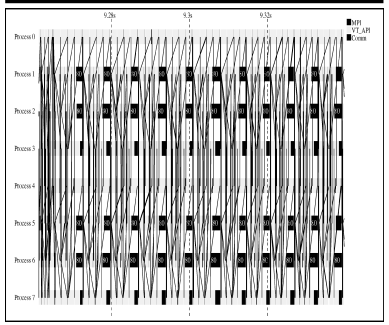
Current HPC architectures

- Moving from transistors to heterogeneity scaling
- Hybrid computing resources: CPUs, GPUs, MICs



Programming hybrid platforms

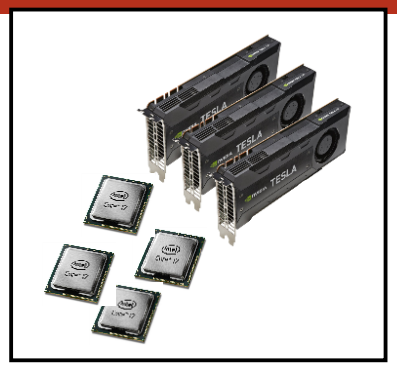
- Traditional, explicit programming models (MPI, CUDA, OpenMP, pthreads, ...)
  - ☺ Perfect control ⇝ maximal achievable performance
  - ☹ Monolithic codes ⇝ hard to develop and maintain
  - ☹ Hard to optimize ⇝ performance portability
  - ☹ Fixed scheduling ⇝ sensitive to variability



Context

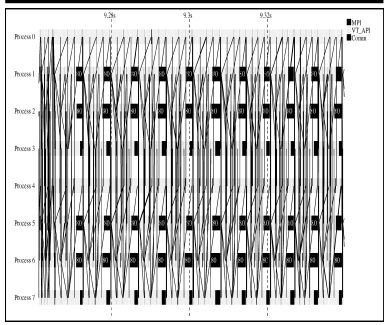
Current HPC architectures

- Moving from transistors to heterogeneity scaling
- Hybrid computing resources: CPUs, GPUs, MICs

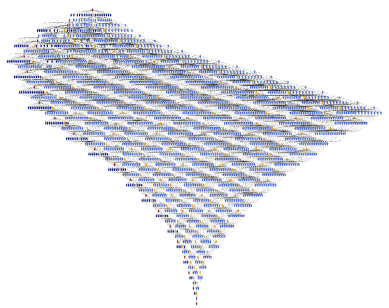


Programming hybrid platforms

- Traditional, explicit programming models (MPI, CUDA, OpenMP, pthreads, ...)
  - ☺ Perfect control $\rightsquigarrow$ maximal achievable performance
  - ☹ Monolithic codes $\rightsquigarrow$ hard to develop and maintain
  - ☹ Hard to optimize $\rightsquigarrow$ performance portability
  - ☹ Fixed scheduling $\rightsquigarrow$ sensitive to variability



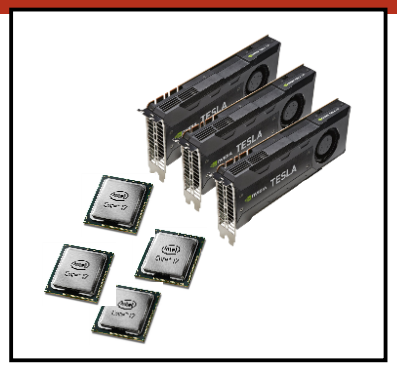
- Recent task-based programming models (PaRSEC, OmpSs, Charm++, StarPU, ...)
  - ☺ Single, abstract programming model based on DAG
  - ☺ Runtime responsible for dynamic scheduling
  - ☺ Portability of code and performance



Context

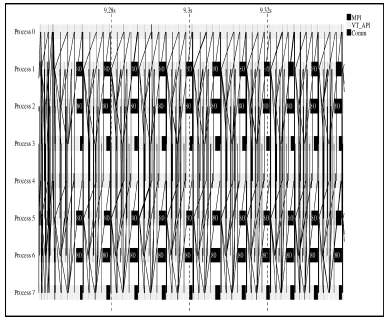
Current HPC architectures

- Moving from transistors to heterogeneity scaling
- Hybrid computing resources: CPUs, GPUs, MICs

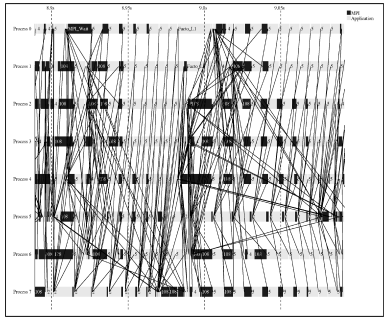


Programming hybrid platforms

- Traditional, explicit programming models (MPI, CUDA, OpenMP, pthreads, ...)
  - ☺ Perfect control ⇝ maximal achievable performance
  - ☹ Monolithic codes ⇝ hard to develop and maintain
  - ☹ Hard to optimize ⇝ performance portability
  - ☹ Fixed scheduling ⇝ sensitive to variability

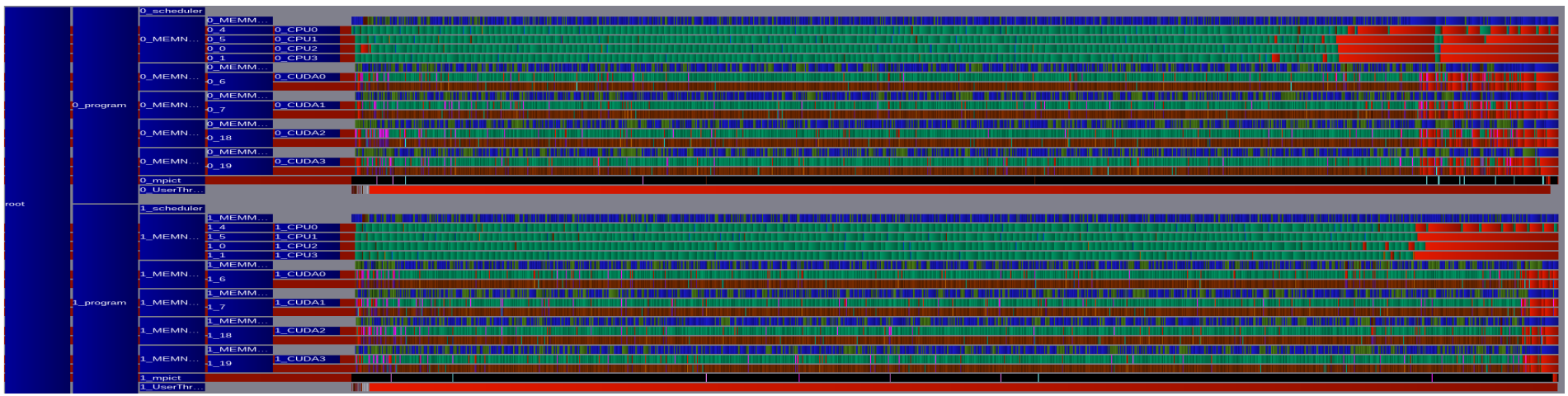


- Recent task-based programming models (PaRSEC, OmpSs, Charm++, StarPU, ...)
  - ☺ Single, abstract programming model based on DAG
  - ☺ Runtime responsible for dynamic scheduling
  - ☺ Portability of code and performance
  - ☹ New challenge ⇝ scheduling heuristic



Visualization of Task Scheduling

Parallel simulation of superscalar scheduling, Haugen, Kurzak, YarKhan, Dongarra. ICPP **2014**.
The QR factorization of a matrix (size: 3960; tiles size: 180)
The QUARK scheduler: 48 cores (one node).

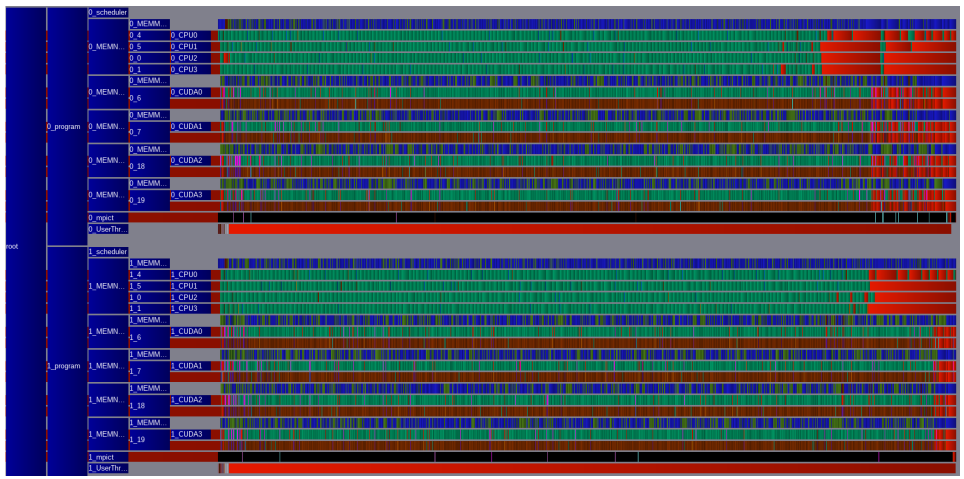


The Cholesky factorization of a matrix (size: 47040; tiles size: 960)
The "MPI-Aware" DMDAS scheduler of StarPU+MPI: 2 nodes with 4 cores and 4 GPUs each.

Related Work: Classical Analysis Tools

Space/time view (resources may be hierarchically organized) + bonus

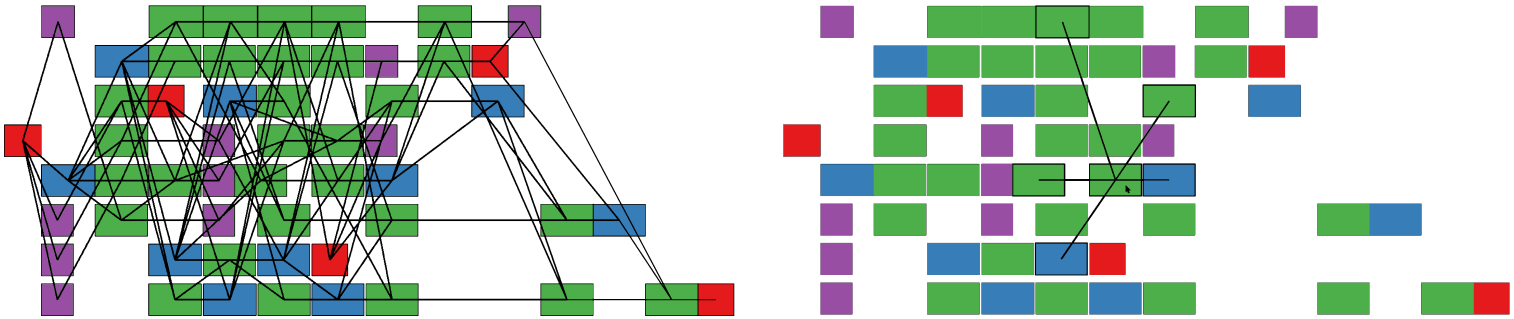
- Paraver (100K) – https://tools.bsc.es/paraver
- Projections (35K) – http://charm.cs.uiuc.edu/software
- FrameSoC (300K+LTTNG) – https://soctrace-inria.github.io/framesoc/
- Ravel (19K) – https://github.com/LLNL/ravel
- Paje (31K in Objective-C) – https://github.com/schnorr/Paje
- ViTE (27K) – http://vite.gforge.inria.fr/



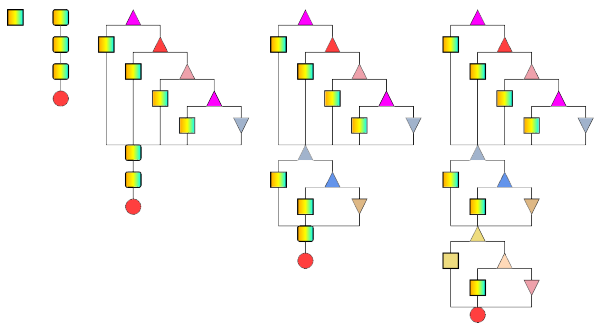
Tiled Cholesky Factorization from StarPU+MPI visualized with ViTE.

Related Work: Emerging Alternatives

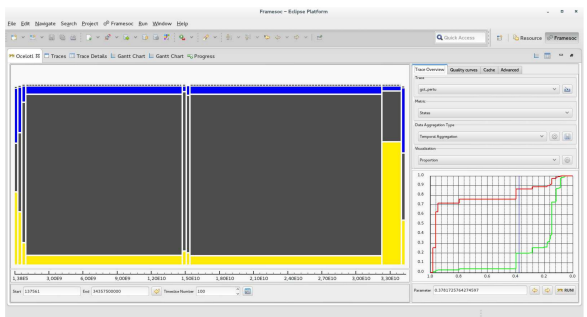
Ad hoc visualization of task dependencies (??? SLOC) *See VPA 2015*



Exploiting DAG structure: DAGViz (??? SLOC) *See VPA 2015*



Entropy-aware aggregation: Ocelotl (3K+300K) https://github.com/soctrace-inria/ocelotl



Current Tools for Visual Performance Analysis Tools

- Implemented in C/C++ to scale
- Interactive (depending on scale) and user friendly (mouse interaction)

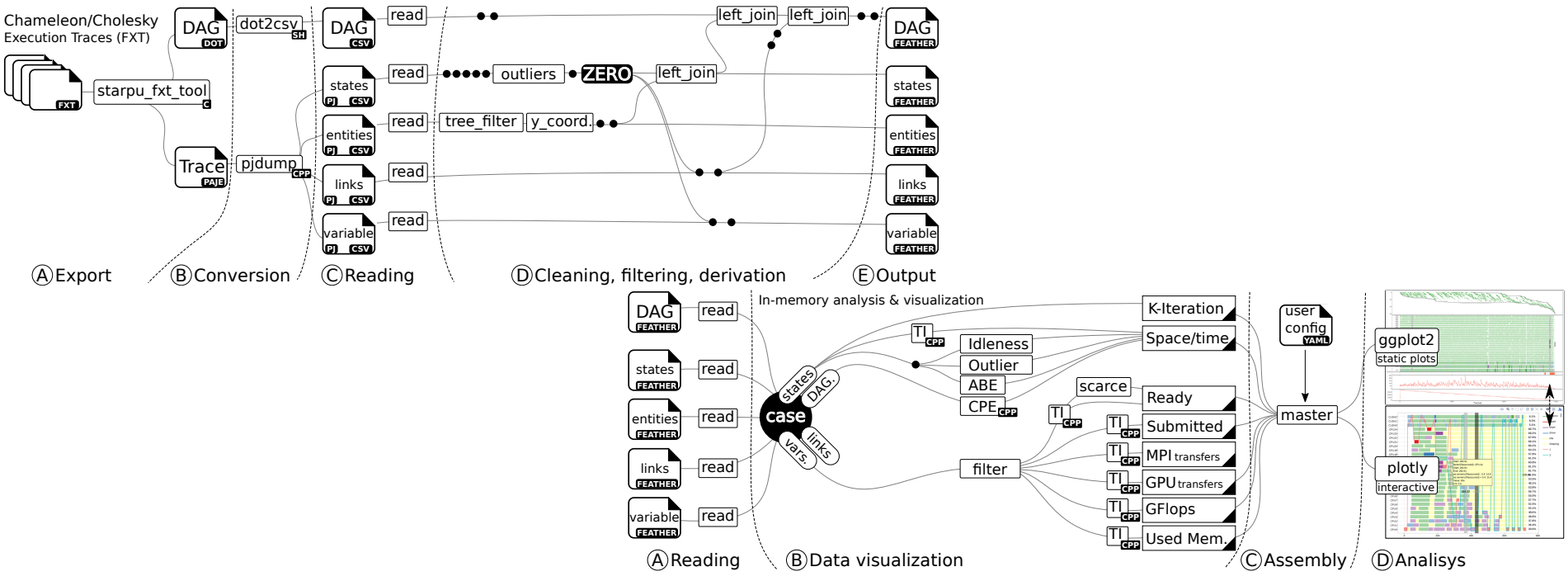
- Large and complex source code, difficult to extend
- Generally not designed for hybrid platforms and dynamic runtimes
- Flexible filter calls for scripting capability
- Lack custom views exploiting application and platform structure

Our (Agile, Scriptable, Flexible) 2-Phase Workflow

Adopt modern data analysis tools for scripting

→ `pj_dump` + `R` + `tidyverse` + `ggplot2` + `plotly` (≈ 3.5K SLOC)

Workflow Execution: screen (1st phase) + org-mode (2nd phase)



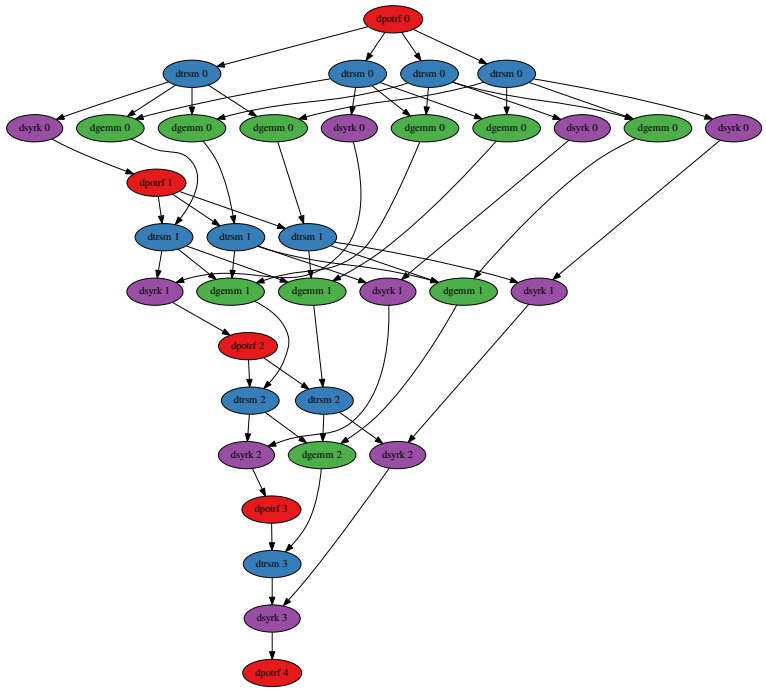
Simplified 2-phase workflow (see our forthcoming paper).

- Fail fast if an idea does not work
- Workflow can be shared to reproduce (and change) the analysis

Experimental validation: application and platform

MORSE – Matrices Over Runtime Systems @ Exascale
http://icl.cs.utk.edu/projectsdev/morse/
Tiled Cholesky factorization available in Chameleon

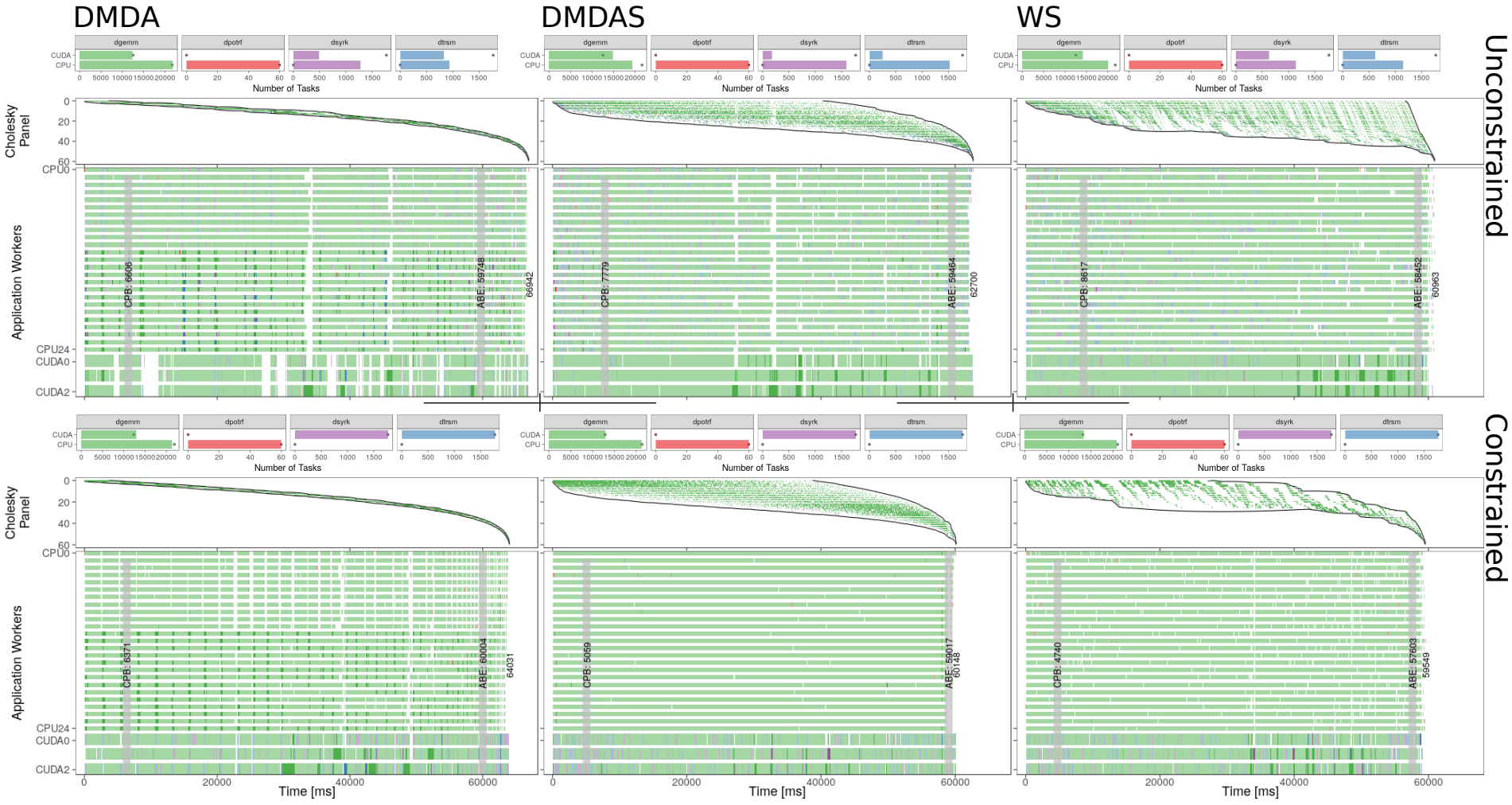
```
for (k = 0; k < N; k++) {
  DPOTRF(RW,A[k][k]);
  for (i = k+1; i < N; i++)
    DTRSM(RW,A[i][k], R,A[k][k]);
  for (i = k+1; i < N; i++) {
    DSYRK(RW,A[i][i], R,A[i][k]);
    for (j = k+1; j < i; j++)
      DGEMM(RW,A[i][j],
       R,A[i][k], R,A[j][k]);
  }
}
```



StarPU runtime on these platforms

- `idcin-2.grenoble.grid5000.fr` (Digitalis, phased out in February 2017)
  - Two 14-core Intel Xeon E5-2697v3 with Three NVIDIA Titan X

Scheduler Comparison (input: 60×60 of 960×960)



Small matrix + interaction (12×12)

→ try yourself at http://perf-ev-runtime.gforge.inria.fr/vpa2016/

Conclusion and Ongoing Work

Achievements

- Flexible analysis workflow in $\approx$ 3.5K SLOC
  - Dynamic task-based applications
  - Multi-node, multi-core, multi-GPU $\cdots$ What's next?
- Suitable for scheduling specialists

Immediate work

- Investigate data dependencies (scheduler) anomalies on scale

Thank you for your attention!

schnorr@inf.ufrgs.br
vgpinto@inf.ufrgs.br

Questions?

Analyzing Dynamic Task-Based Applications on Hybrid Platforms: An Agile Scripting Approach. 3rd Workshop on Visual Performance Analysis (VPA)
https://hal.inria.fr/hal-01353962