Computing fast, computing smart

Luigi Nardi, Ph.D.

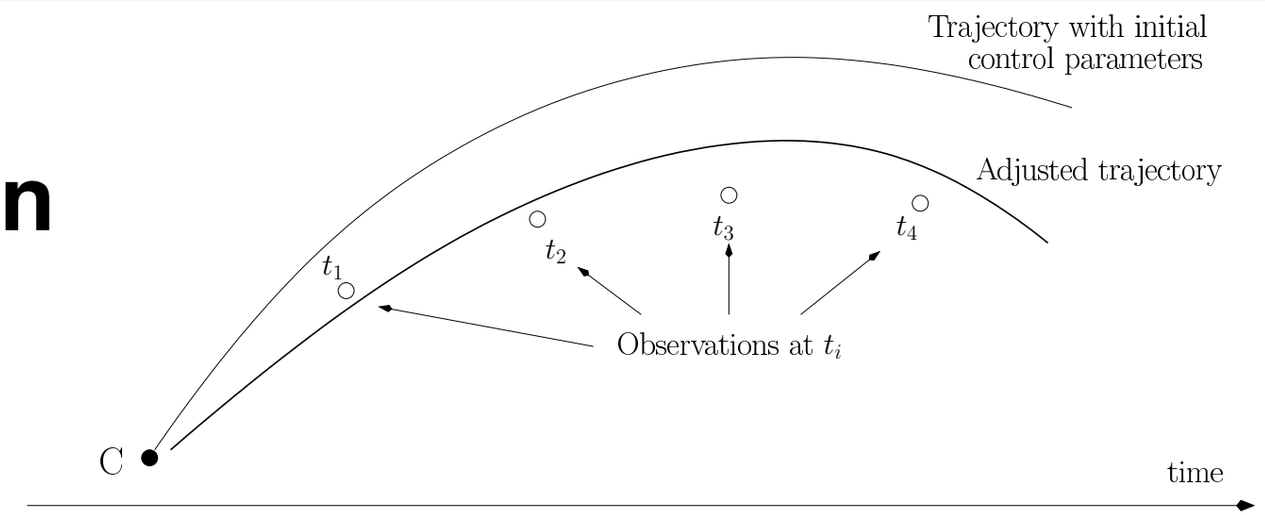
Dep. of Computing, Imperial College London
Software Performance Optimisation group
@ECMWF, May 11th 2015
http://www.doc.ic.ac.uk/~lnardi

Imperial College London

EPSRC
Engineering and Physical Sciences Research Council

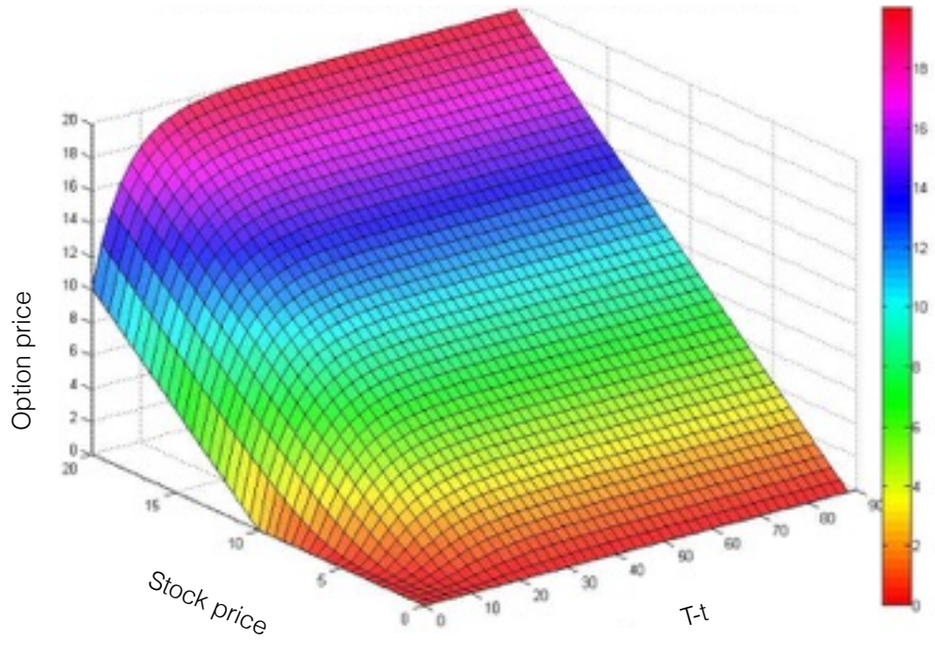
Three application domains

1) Variational data assimilation



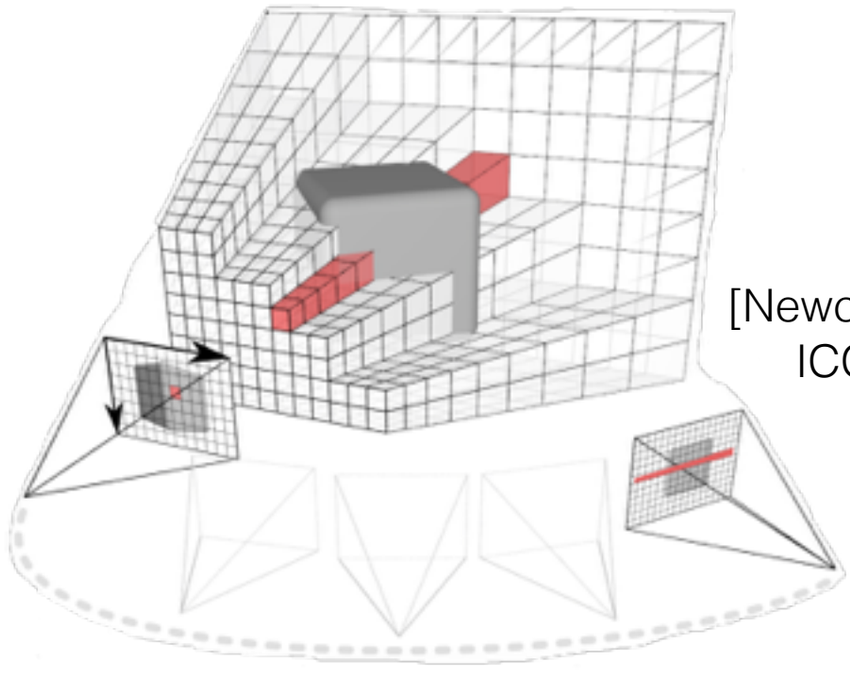
Ph.D. Computer/computational science, LOCEAN and CEDRIC labs

2) Computational finance



Permanent researcher, R&D at Murex S.A.S.

3) Computer vision



[Newcombe et al., ICCV 2011]

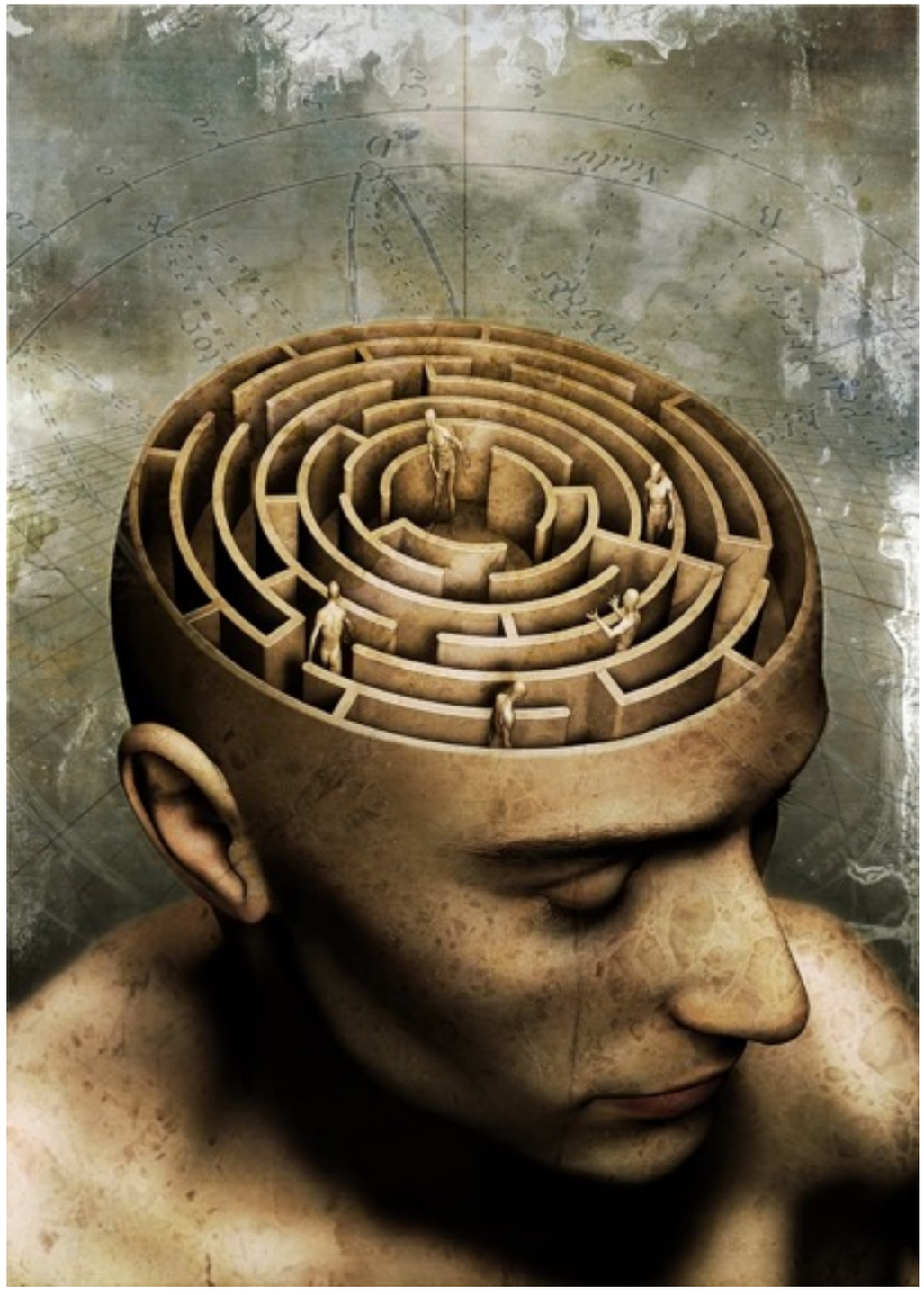
Research Associate, ICL (Prof. Paul Kelly)
Software Performance Optimisation group

Outline

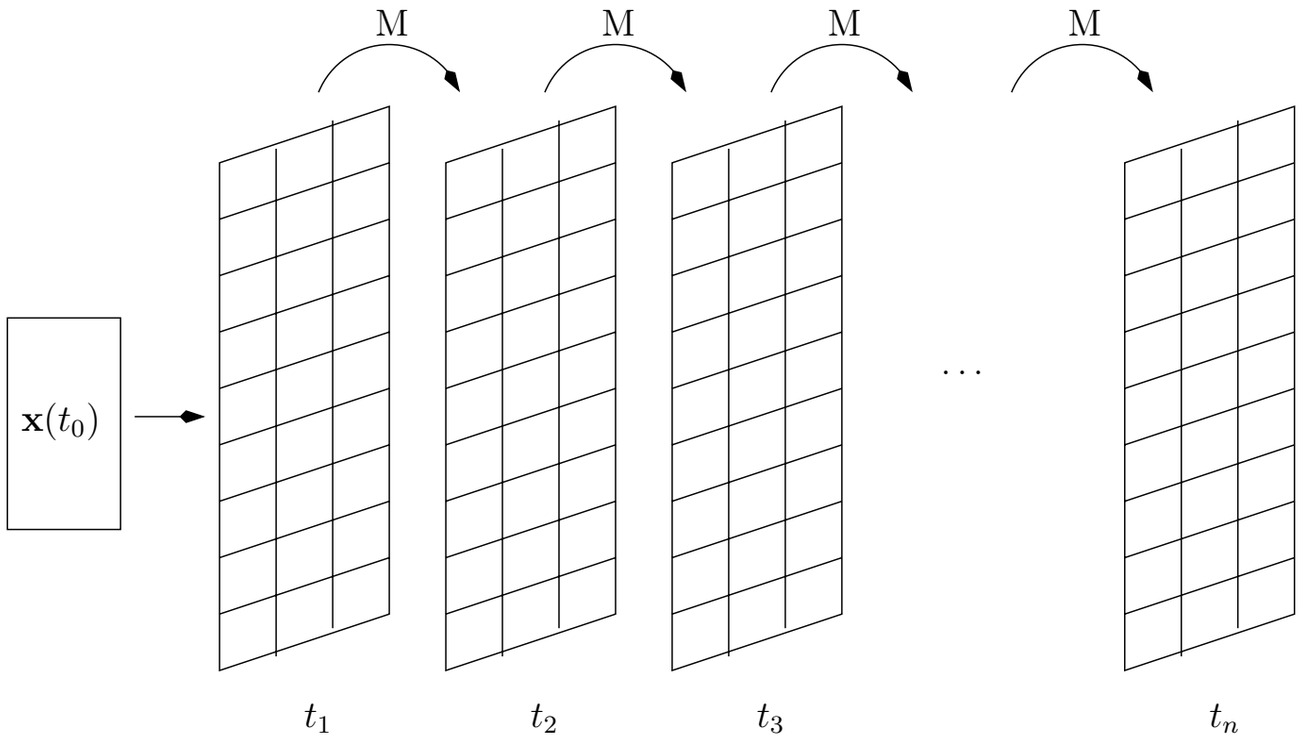
- **Domain-specific languages, a practical use case**

- Micro-clusters of ARMs and GPUs

- A holistic approach to performance and optimisation



Direct numerical model

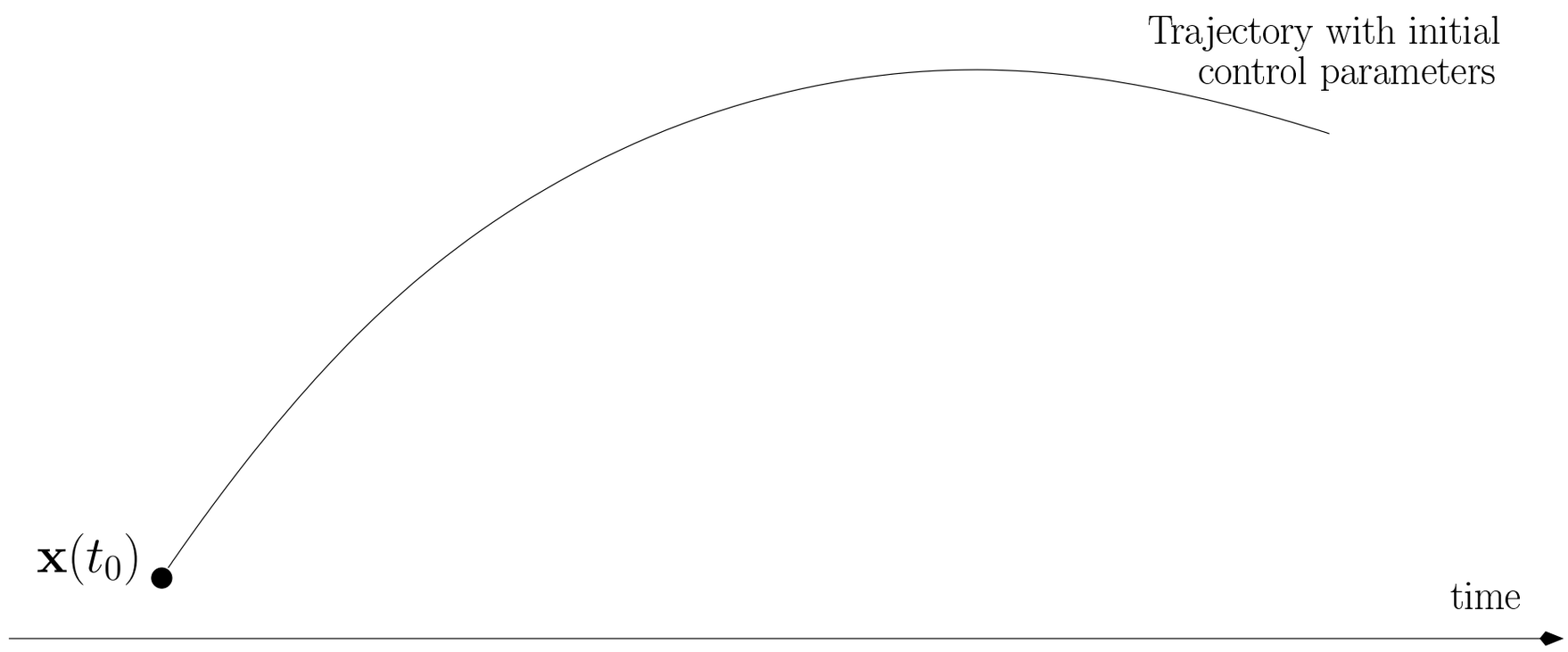


- Structured mesh
- M: direct model between two time steps $t_i,\ t_{i+1}$
- $\mathbf{x}(t_0)$: control parameters
- $M_i(\mathbf{x}(t_0))$: state at t_i starting from state at t_0

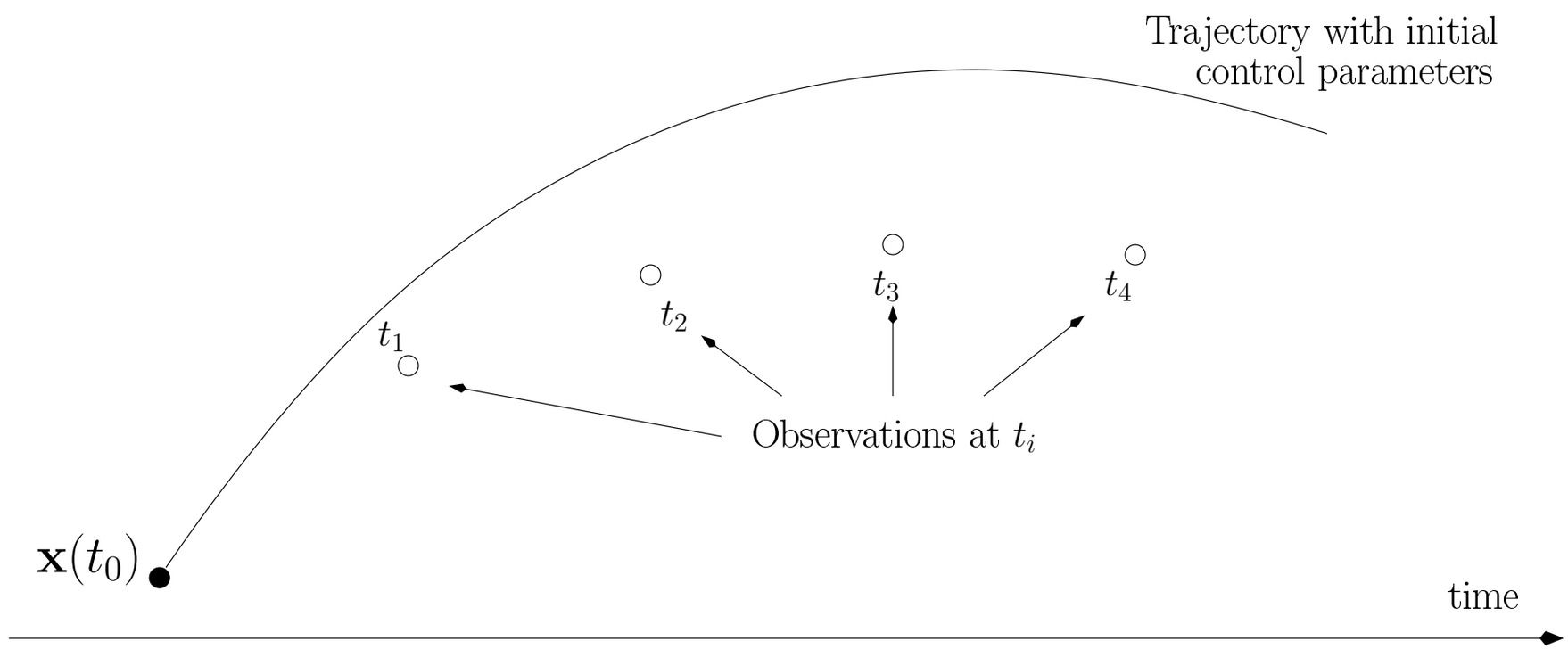
Models used to forecast or analyse the evolution of phenomena.

Models are not perfect

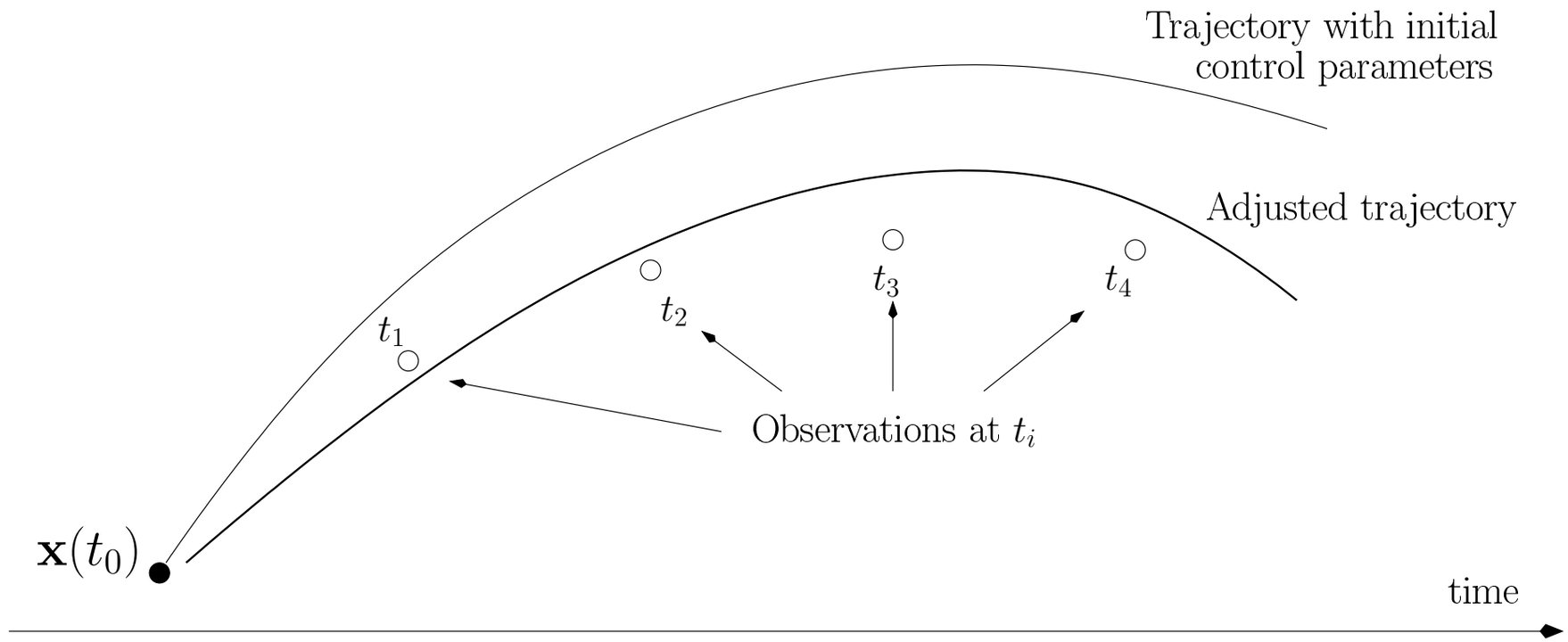
Variational data assimilation 4D-Var



Variational data assimilation 4D-Var



Variational data assimilation 4D-Var



Variational DA formalism

Cost function:

$$J(\mathbf{x}(t_0)) = \underbrace{\frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)}_{data\ \ fitting} + \underbrace{\frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b)}_{regularisation}$$

- $\mathbf{y}_i$: model output
- $\mathbf{y}_i^o$: observations at time t_i
- $\mathbf{R}_i, \mathbf{B}$: variance-covariance matrices
- $\mathbf{x}^b$: background vector

Variational DA formalism

Cost function:

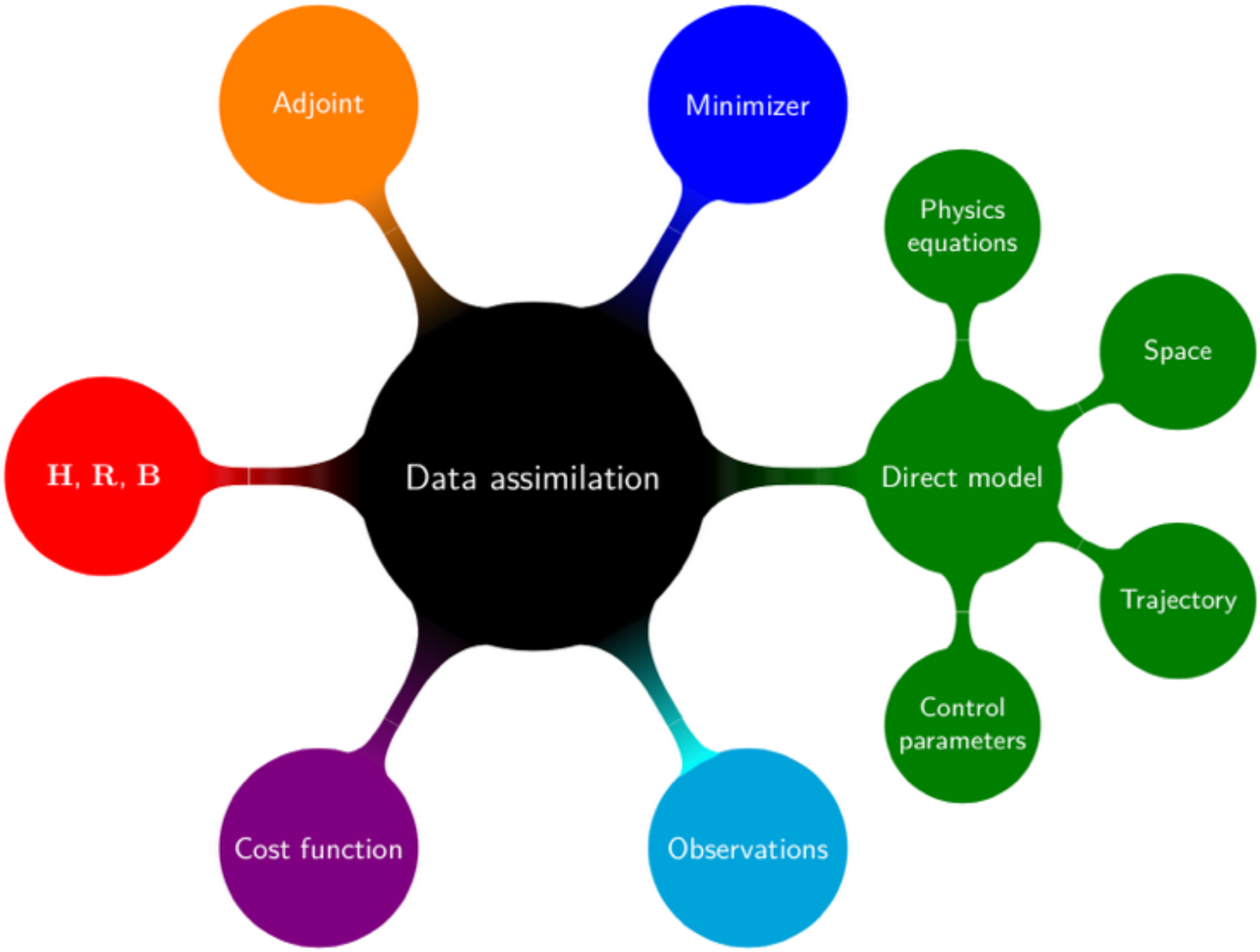
$$J(\mathbf{x}(t_0)) = \underbrace{\frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)}_{data\ \ fitting} + \underbrace{\frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b)}_{regularisation}$$

- $\mathbf{y}_i$: model output
- $\mathbf{y}_i^o$: observations at time t_i
- $\mathbf{R}_i, \mathbf{B}$: variance-covariance matrices
- $\mathbf{x}^b$: background vector

$$\nabla_{\mathbf{x}(t_0)} J = \sum_{i=1}^{n} \underbrace{\mathbf{M}_i^T(\mathbf{x}(t_0))}_{adjoint}\mathbf{H}_i^T[\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)] + \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b)$$

- H_i: observation operator
- $\mathbf{M}_i^T(\mathbf{x}(t_0))$: adjoint model

Variational DA components



Why DA is hard

1. Adjoint model hard to implement

Why DA is hard

1. Adjoint model hard to implement

2. New improvements/advances in DA hard to test

Why DA is hard

1. Adjoint model hard to implement
2. New improvements/advances in DA hard to test
3. Big data. ECMWF example:
   - Observations per day order of 10^7
   - State vector order of 10^9

Why DA is hard

1. Adjoint model hard to implement
2. New improvements/advances in DA hard to test
3. Big data. ECMWF example:
   - Observations per day order of 10^7
   - State vector order of 10^9
4. “Performance" optimisation and trade-off:
   - Runtime (under target time constraint)
   - Energy (under target energy budget)
   - Accuracy
   - Memory consumption

Software for Variational DA

1. Automatic differentiators: TAPENADE, TAF, OpenAD
2. Minimisers: M1QN3, M2QN1, TAO
3. Model coupling: OASIS, PALM
4. Performance: OpenMP, MPI, OpenCL, CUDA

Software for Variational DA

1. Automatic differentiators: TAPENADE, TAF, OpenAD
2. Minimisers: M1QN3, M2QN1, TAO
3. Model coupling: OASIS, PALM
4. Performance: OpenMP, MPI, OpenCL, CUDA

Is it possible to ease the DA development?

Domain-specific languages (DSLs)

- DSLs are a hot topic
- Value comes from:
  - Expressivity
  - Performance optimisation
  - Separation of concerns
- Barriers for adoption: toolchain support, debug, profile, tools

Domain-specific languages (DSLs)

- DSLs are a hot topic
- Value comes from:
  - Expressivity
  - Performance optimisation
  - Separation of concerns
- Barriers for adoption: toolchain support, debug, profile, tools

Not all DSLs are languages:

- DSLs can be embedded in a host language
- Libraries are the original DSL mechanism
- Active libraries

Domain-specific languages (DSLs)

- DSLs are a hot topic
- Value comes from:
  - Expressivity
  - Performance optimisation
  - Separation of concerns
- Barriers for adoption: toolchain support, debug, profile, tools

Not all DSLs are domain-specific:

- Some are concern-specific, some are market-specific, some are sometimes called Performance DSLs
- Linear algebra: Matlab, LGen
- Performance: OP2, **PyOP2**, OPS, OpenMP, OpenACC, CUDA, OpenCL, CPN, MARE, SCoP
- Stencil: Pochoir, PLUTO

Not all DSLs are languages:

- DSLs can be embedded in a host language
- Libraries are the original DSL mechanism
- Active libraries

Domain-specific languages (DSLs)

- DSLs are a hot topic
- Value comes from:
  - Expressivity
  - Performance optimisation
  - Separation of concerns
- Barriers for adoption: toolchain support, debug, profile, tools

Not all DSLs are domain-specific:

- Some are concern-specific, some are market-specific, some are sometimes called Performance DSLs
- Linear algebra: Matlab, LGen
- Performance: OP2, **PyOP2**, OPS, OpenMP, OpenACC, CUDA, OpenCL, CPN, MARE, SCoP
- Stencil: Pochoir, PLUTO

Not all DSLs are languages:

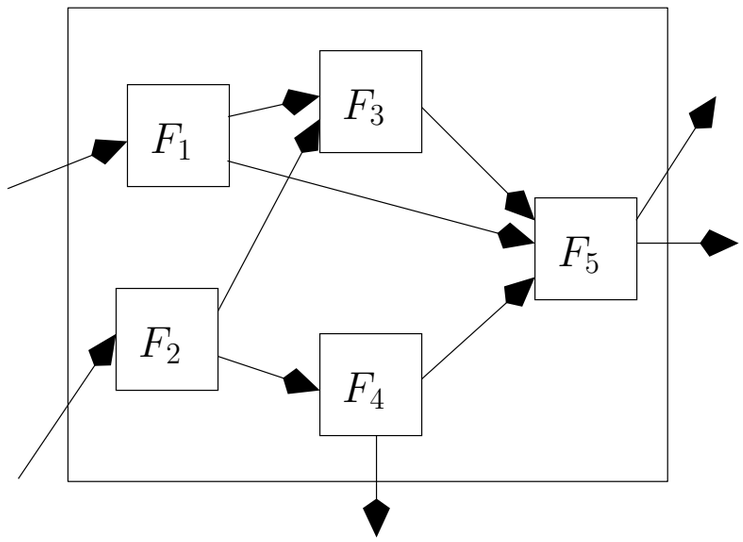
- DSLs can be embedded in a host language
- Libraries are the original DSL mechanism
- Active libraries

Examples of domain languages:

- Geophysics: **YAO**, **Firedrake**, PyFR, FEniCS, ICON DSL
- Finance: **MACS**
- Image proc.: Halide, Darkroom, HiPACC, RIPL
- Graphics: Picture
- Music: Faust, OpenMusic
- Artificial intelligence: Prolog

YAO DSL

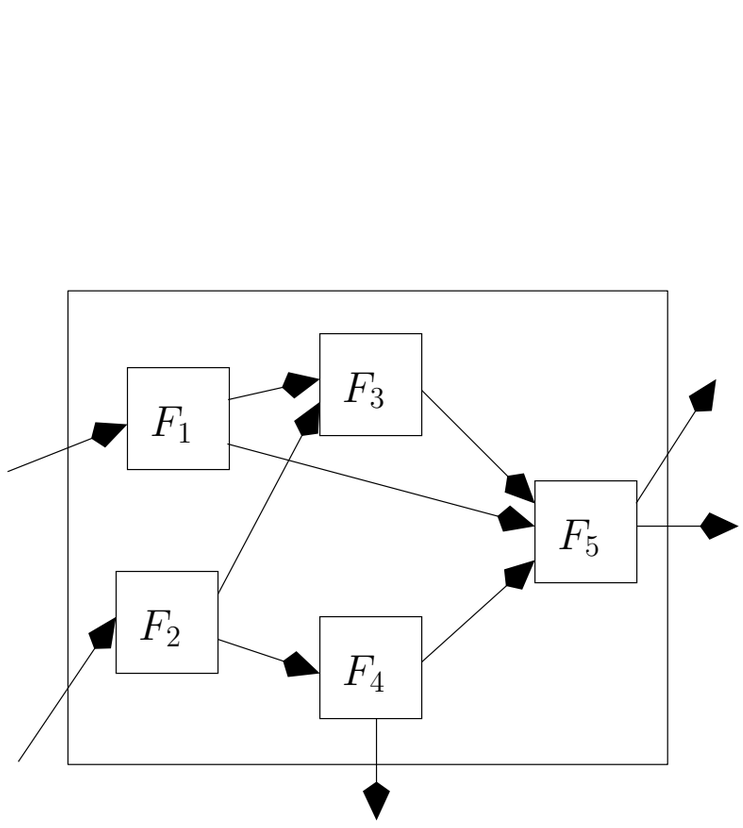
- Based on the modular graph concept (inspired by Neural Networks)
- The same graph is repeated in space and time

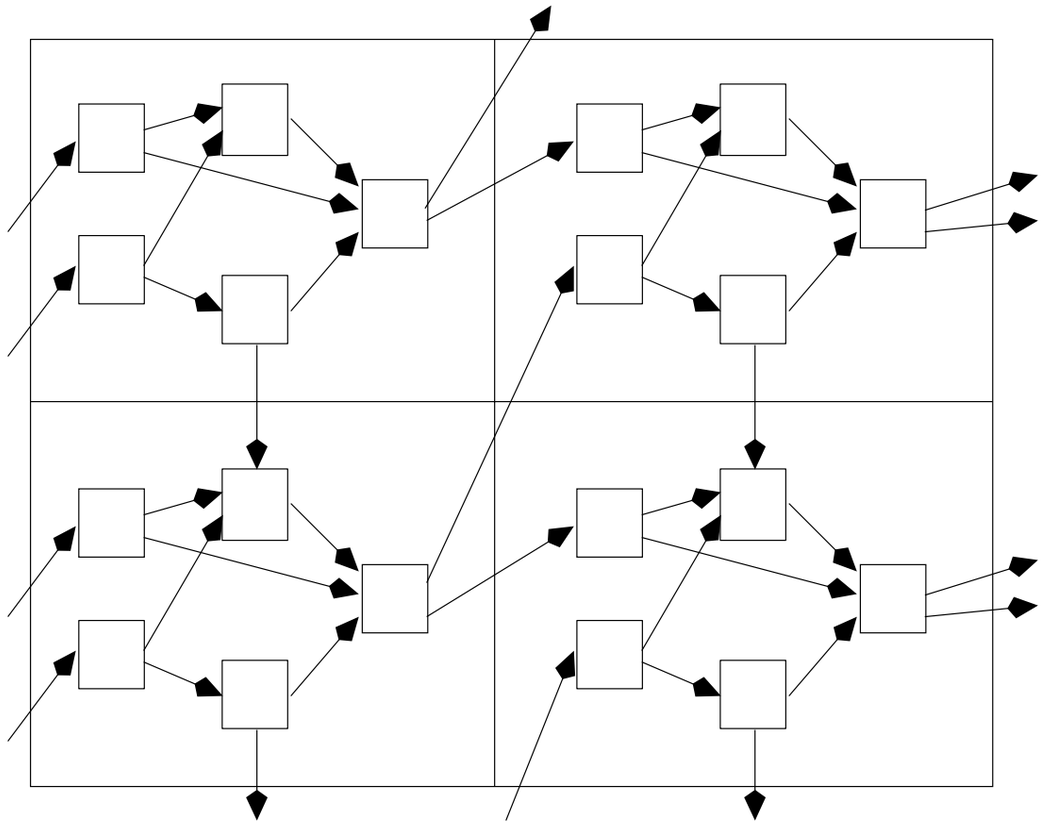


[Nardi et al., 2009]

YAO DSL

- Based on the modular graph concept (inspired by Neural Networks)
- The same graph is repeated in space and time

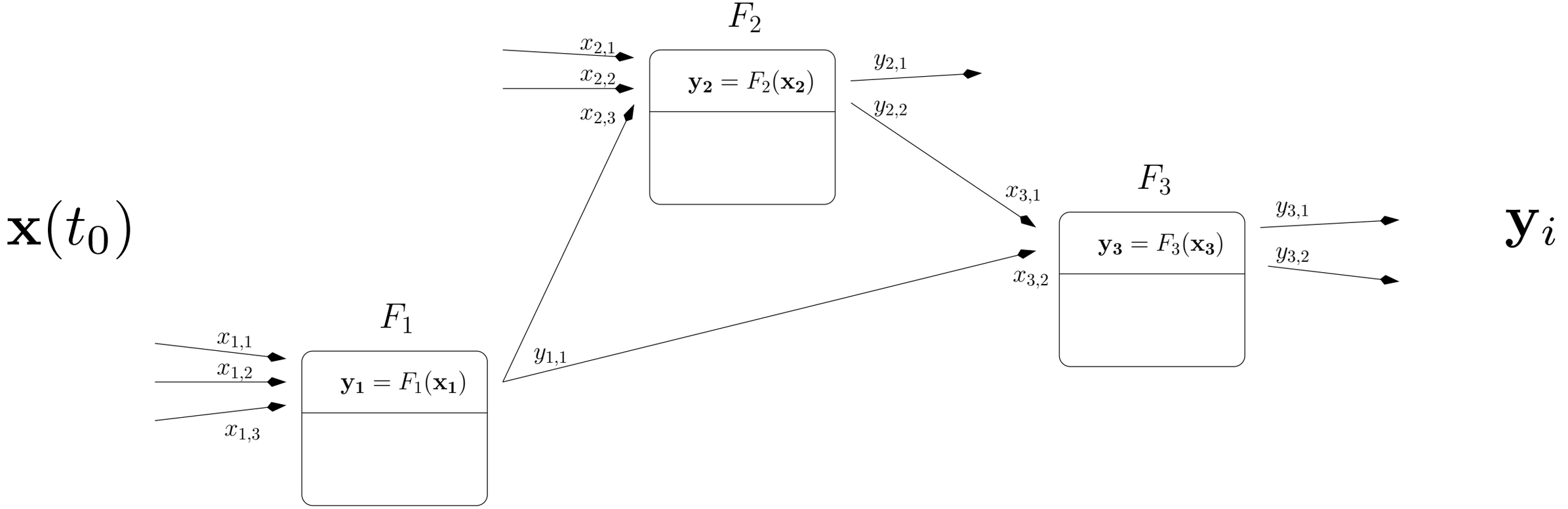




[Nardi et al., 2009]

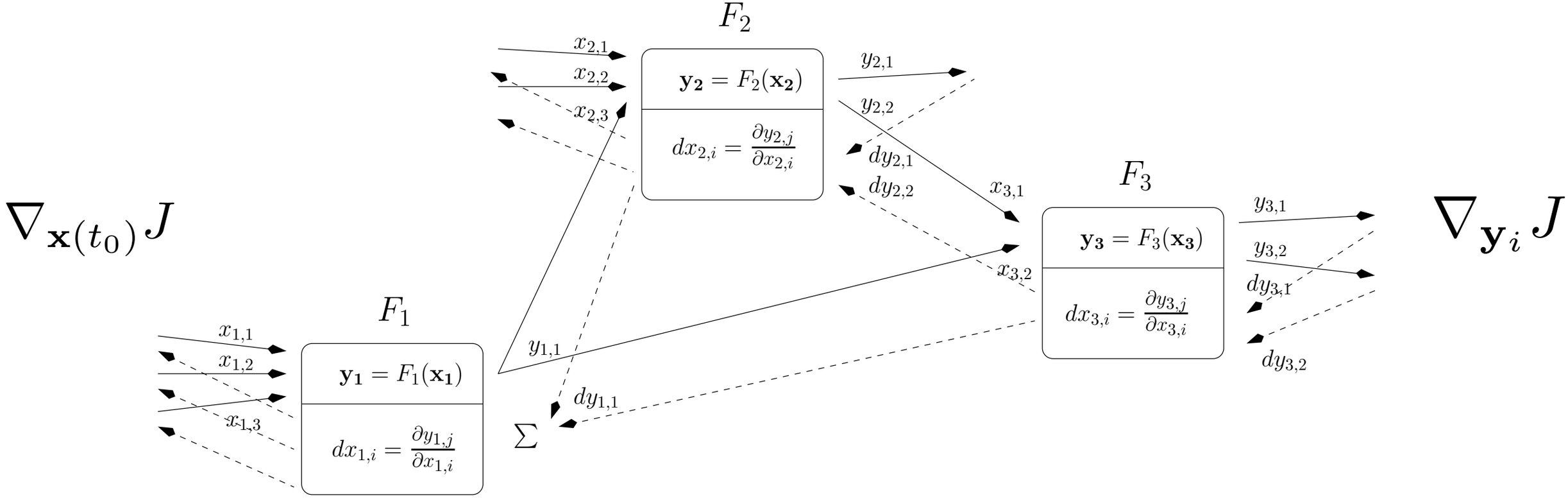
Direct and adjoint models

- In a topological order computes the model:
  forward algorithm

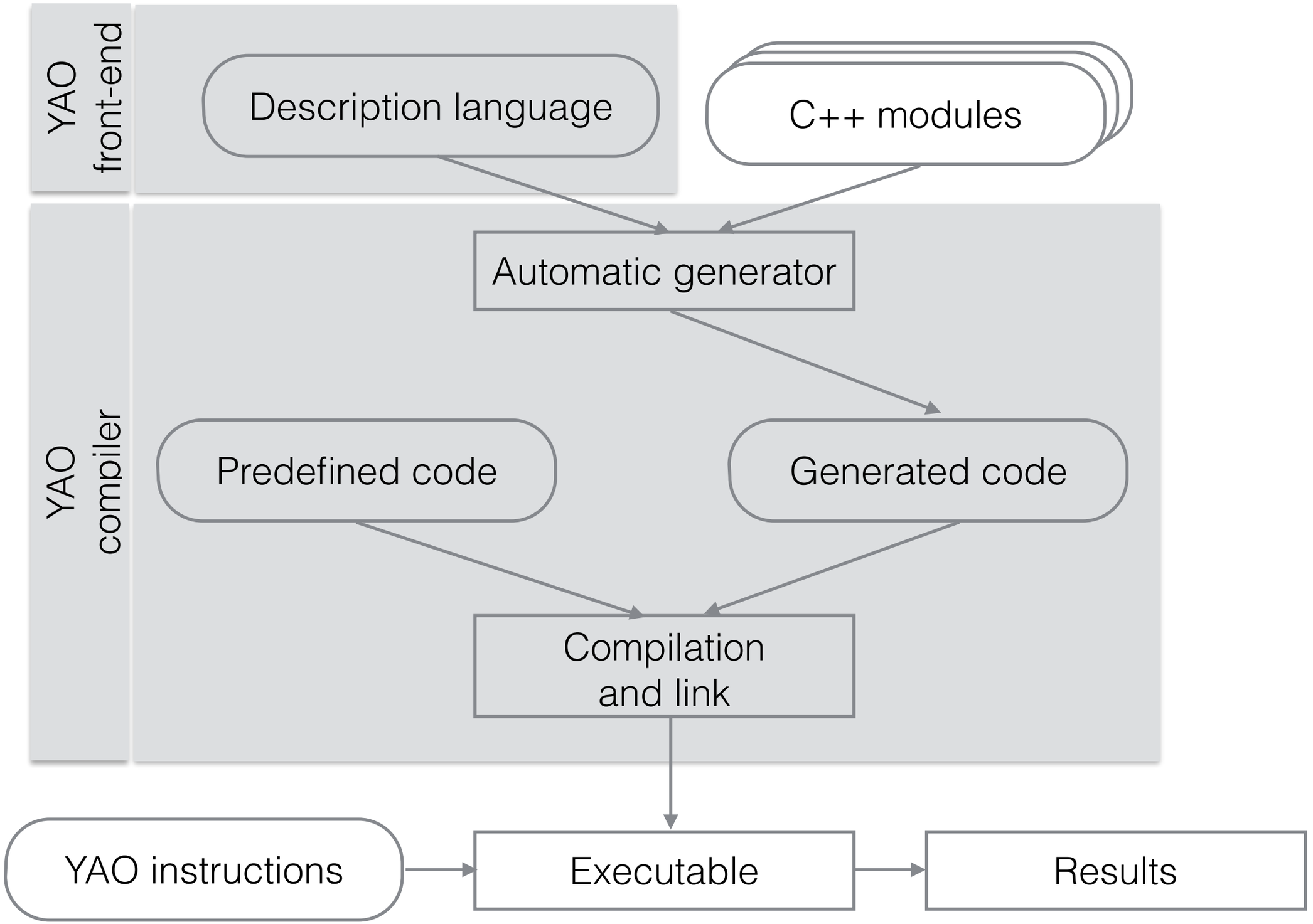


Direct and adjoint models

- In a topological order computes the model:
  forward algorithm
- In a reverse topological order computes the adjoint model:
  backward algorithm

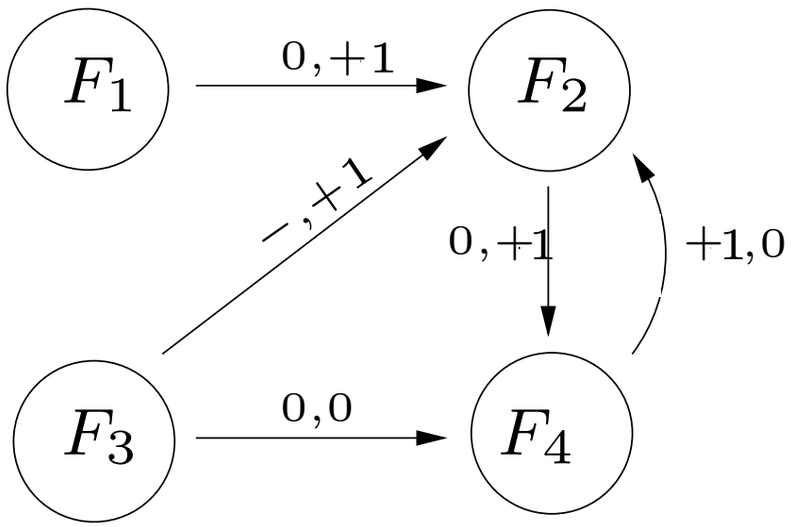


YAO architecture



Automatic generation of parallel code

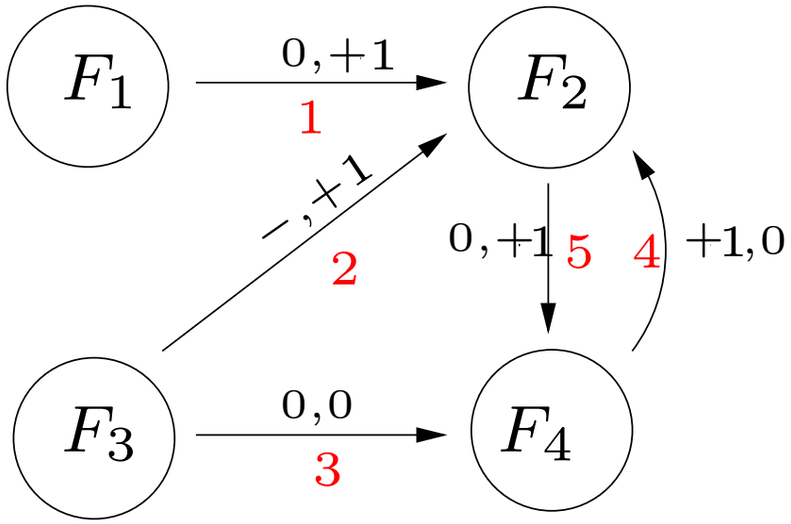
- YAO graph similar to Reduced Dependence Graph with explicit distance vectors
- This similarity enables existing techniques adaptation

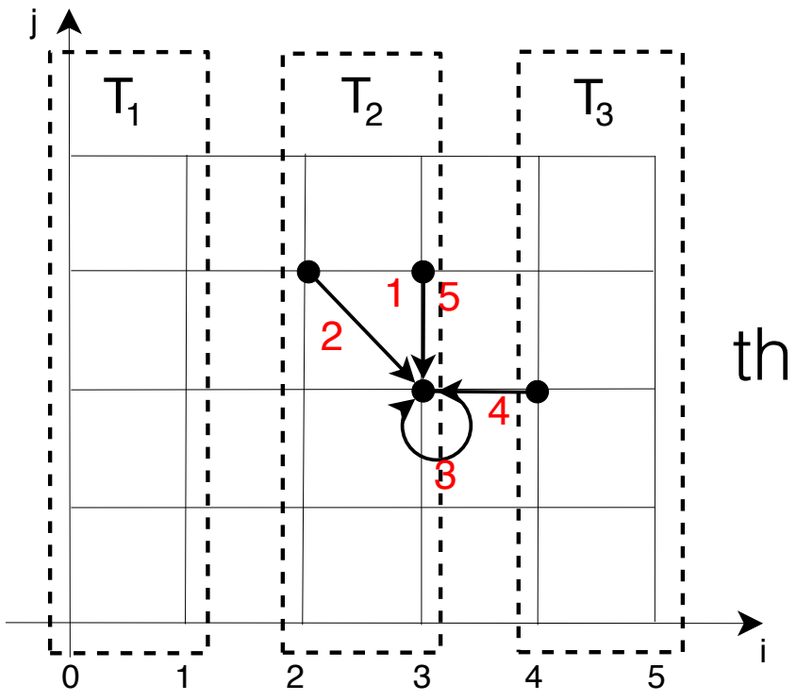


[Nardi et al., 2012]

Automatic generation of parallel code

- YAO graph similar to Reduced Dependence Graph with explicit distance vectors
- This similarity enables existing techniques adaptation



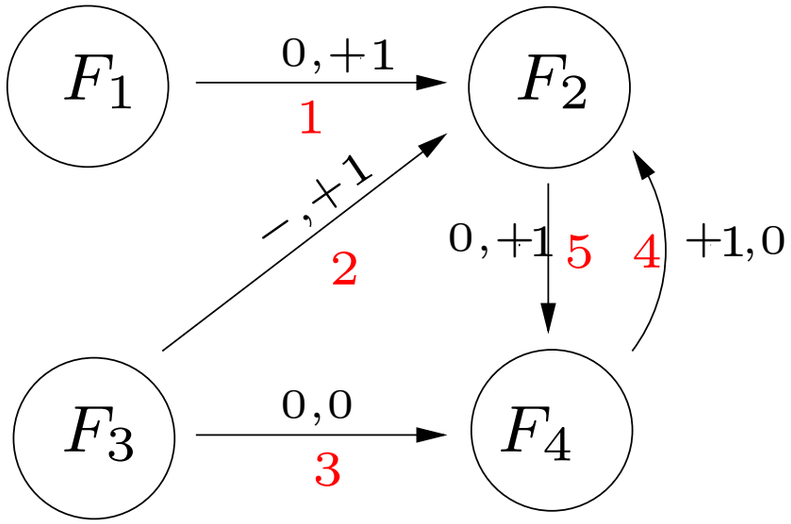


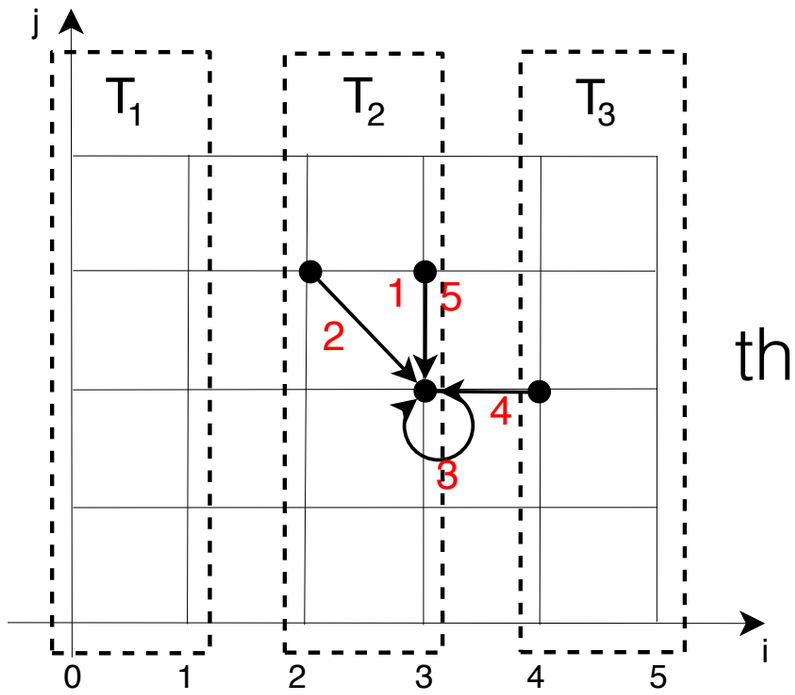
T_1, T_2, T_3: threads 1, 2, 3

[Nardi et al., 2012]

Automatic generation of parallel code

- YAO graph similar to Reduced Dependence Graph with explicit distance vectors
- This similarity enables existing techniques adaptation





T_1, T_2, T_3: threads 1, 2, 3

Parallelisation algorithm:

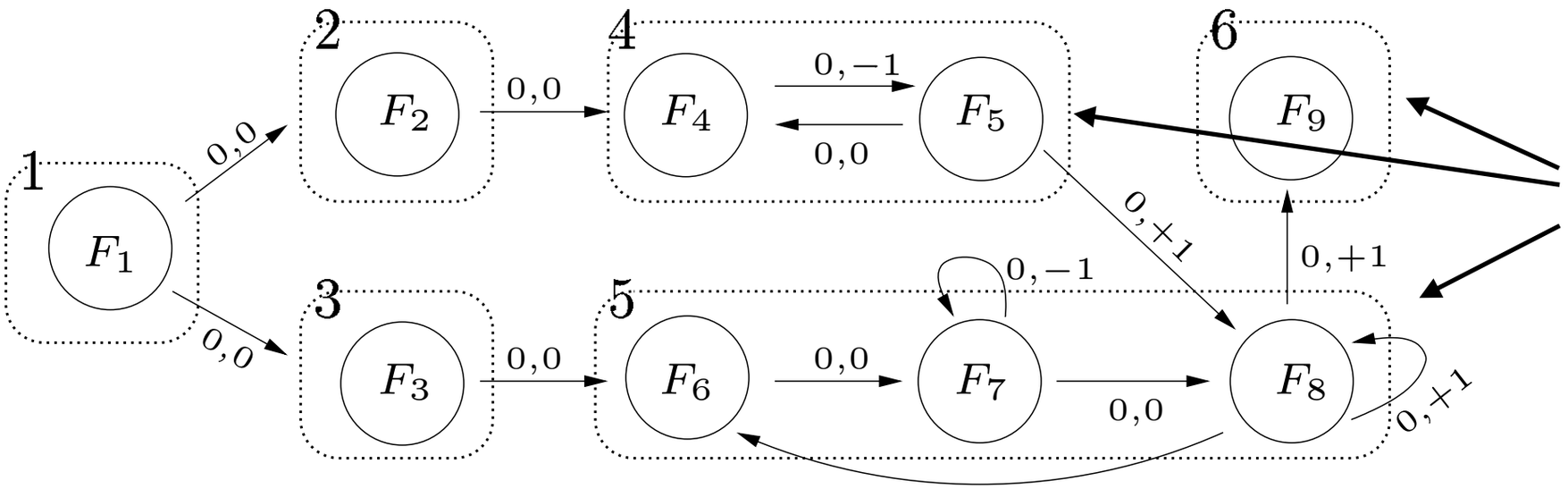
- Maximum loop distribution (Allen-Kennedy)
- Loop fusion (Kennedy-McKinley variant)
- Automatic generation of OpenMP directives

[Nardi et al., 2012]

Automatic parallelisation example

Marine acoustics: variational inversion of sound speed profile and retrieval of geoacoustic parameters (celerity, density, attenuation)

[Berrada et al., 2008]

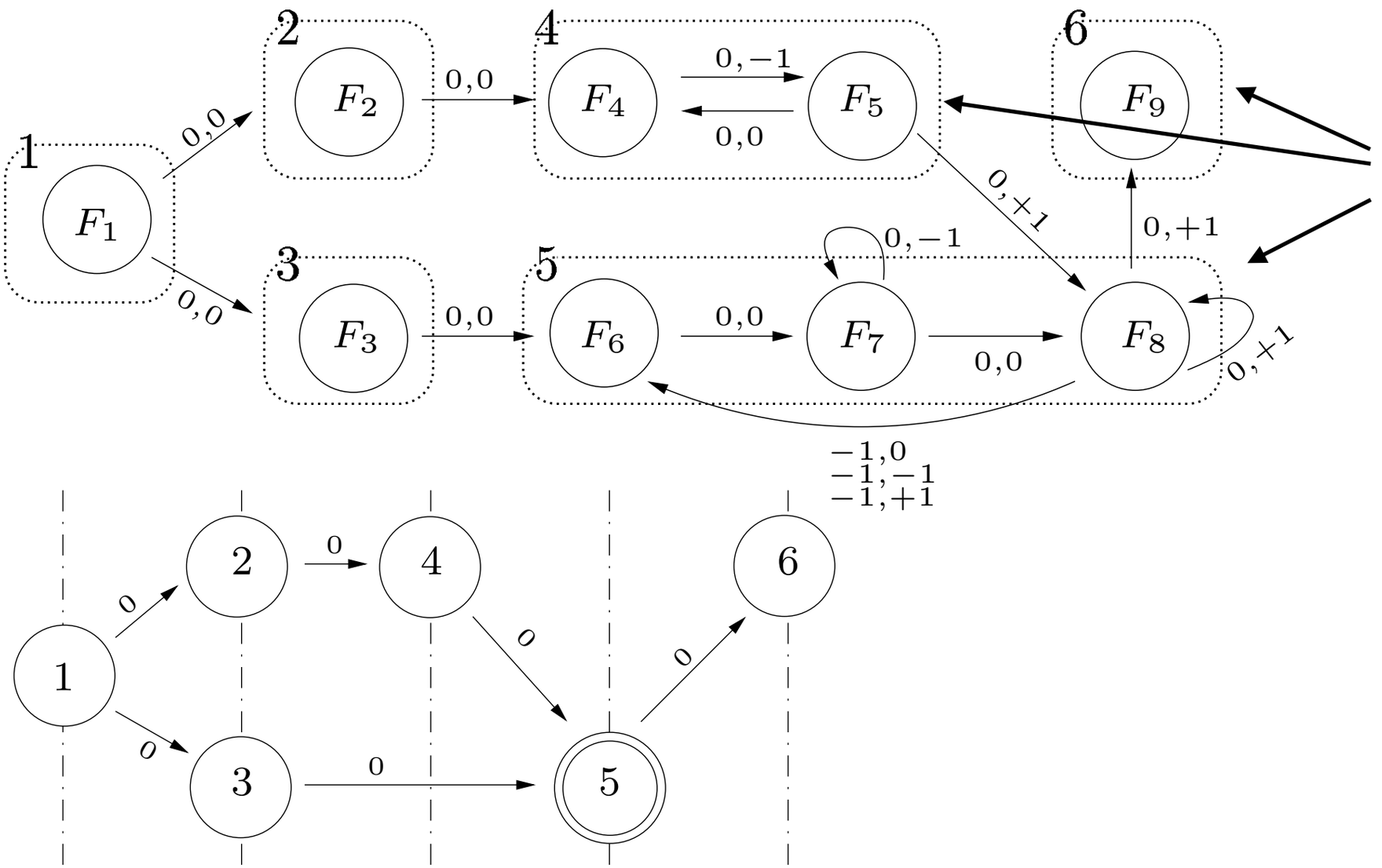


Strongly connected components

Automatic parallelisation example

Marine acoustics: variational inversion of sound speed profile and retrieval of geoacoustic parameters (celerity, density, attenuation)

[Berrada et al., 2008]



Strongly connected components

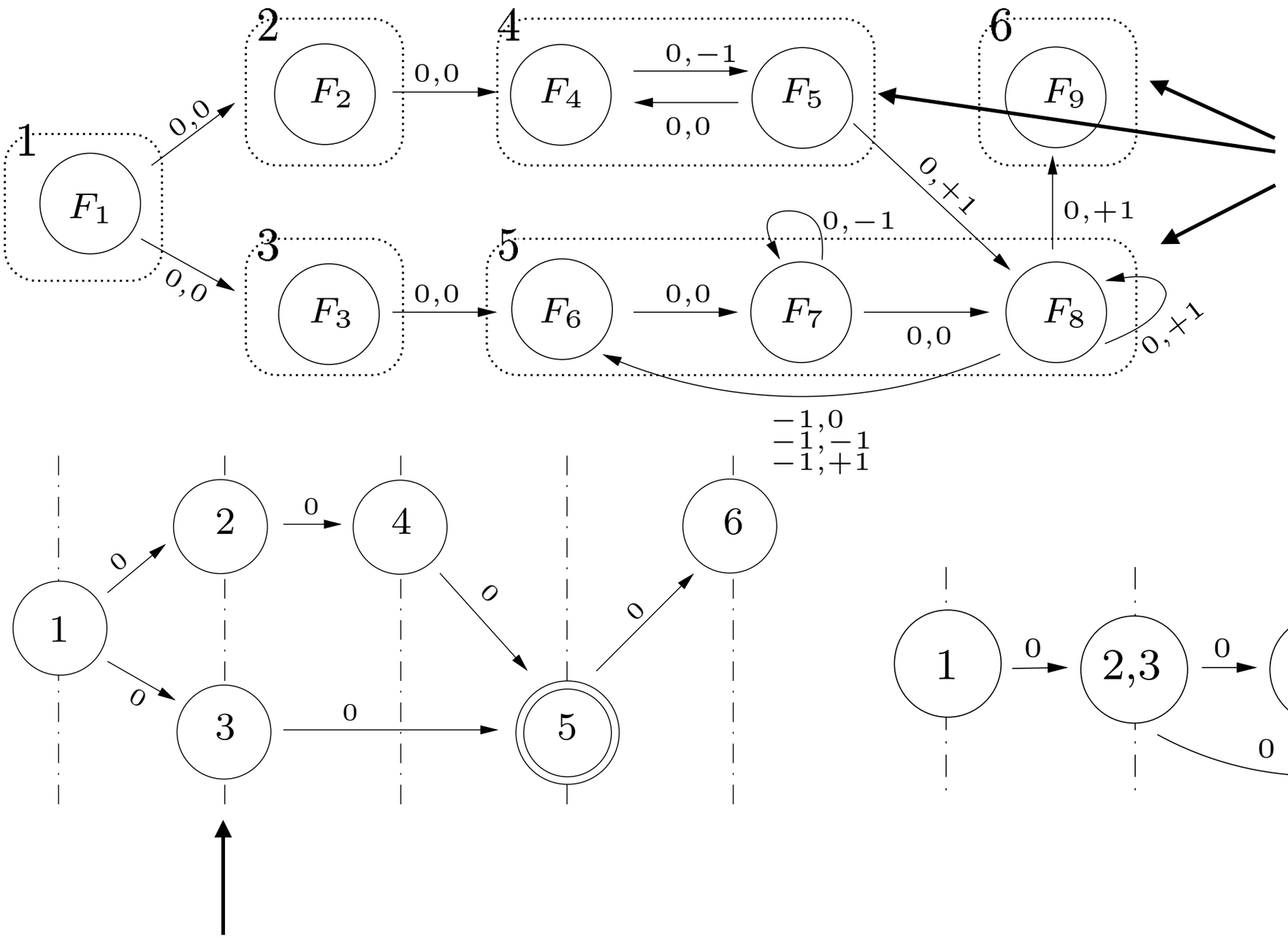
Non parallel:

Parallel:

Automatic parallelisation example

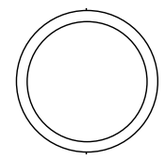
Marine acoustics: variational inversion of sound speed profile and retrieval of geoacoustic parameters (celerity, density, attenuation)

[Berrada et al., 2008]

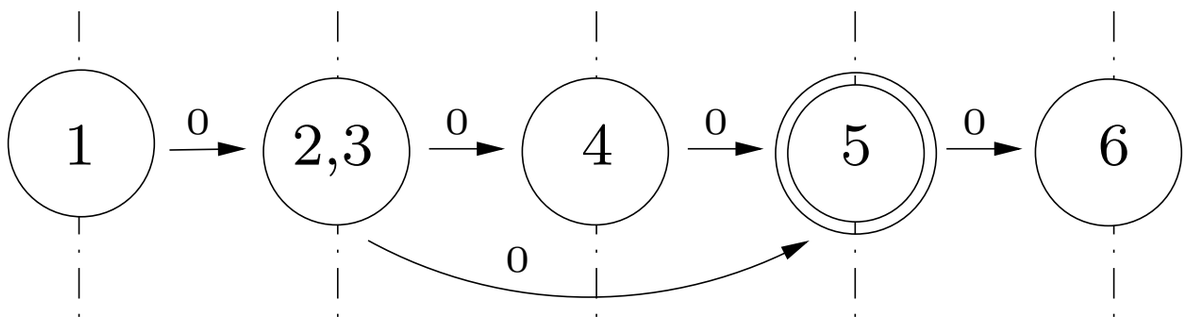


Strongly connected components

Non parallel:



Parallel:

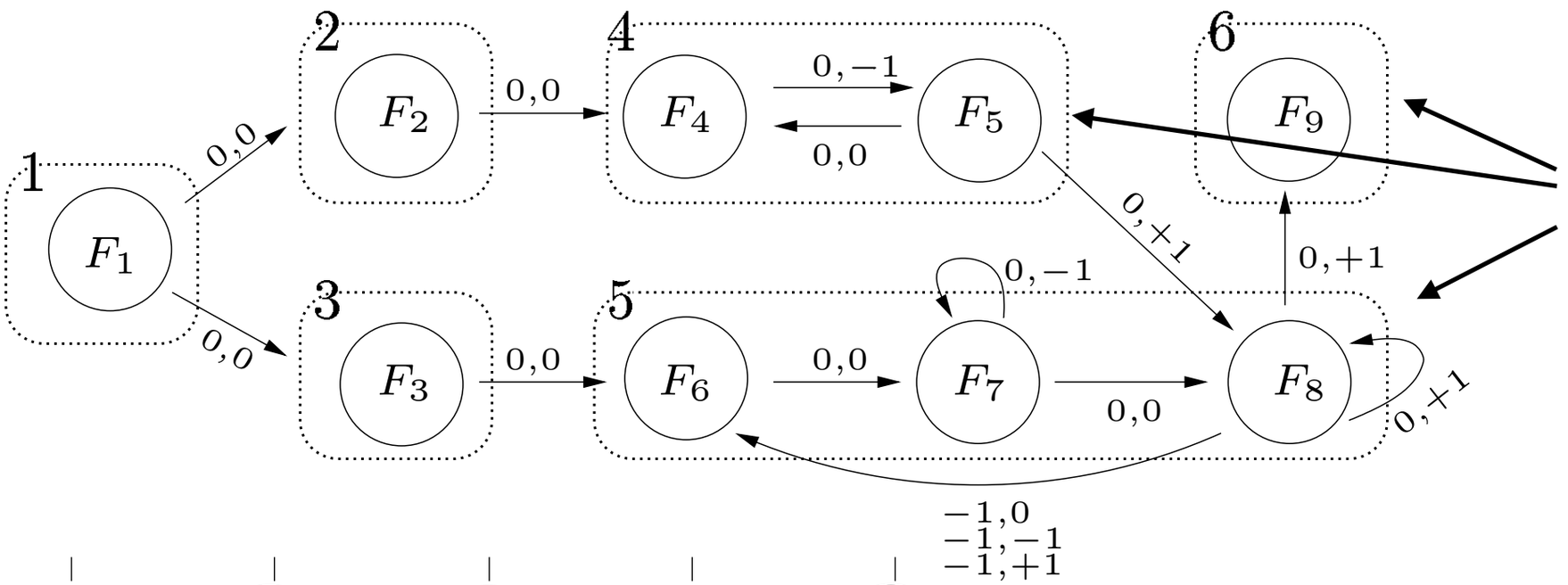


Fusion on levels first

Automatic parallelisation example

Marine acoustics: variational inversion of sound speed profile and retrieval of geoacoustic parameters (celerity, density, attenuation)

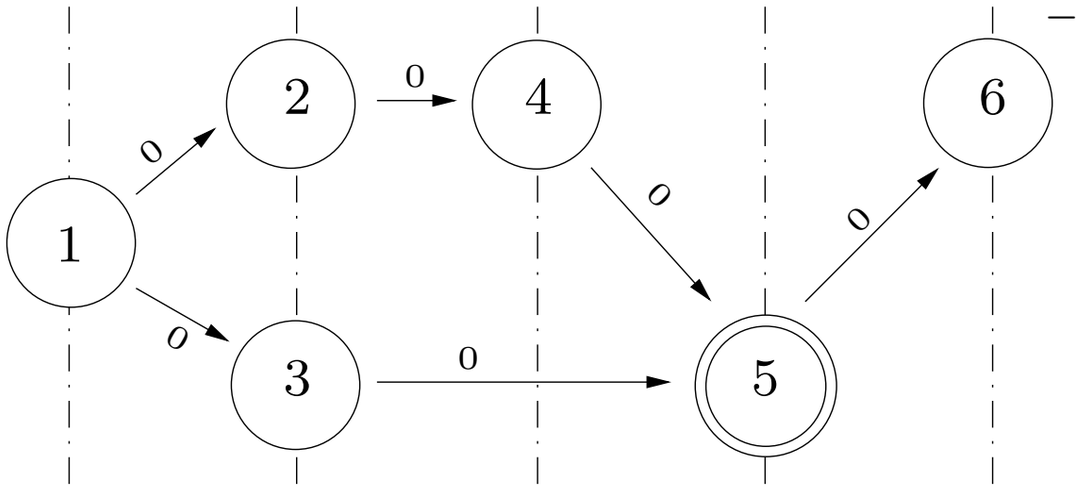
[Berrada et al., 2008]



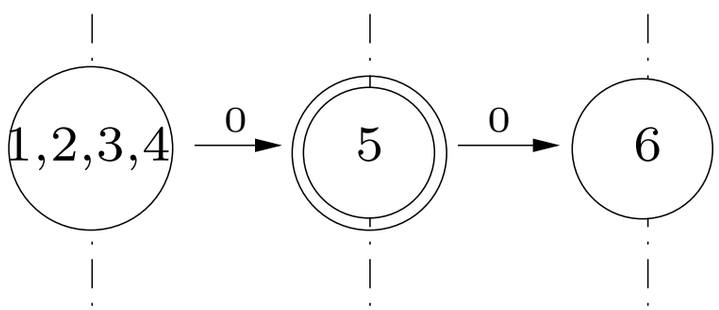
Strongly connected components

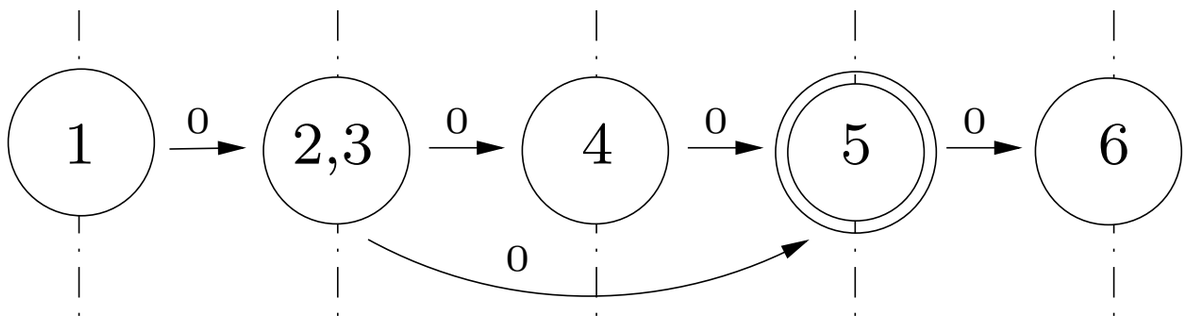
Non parallel:

Parallel:



Fusion on levels first





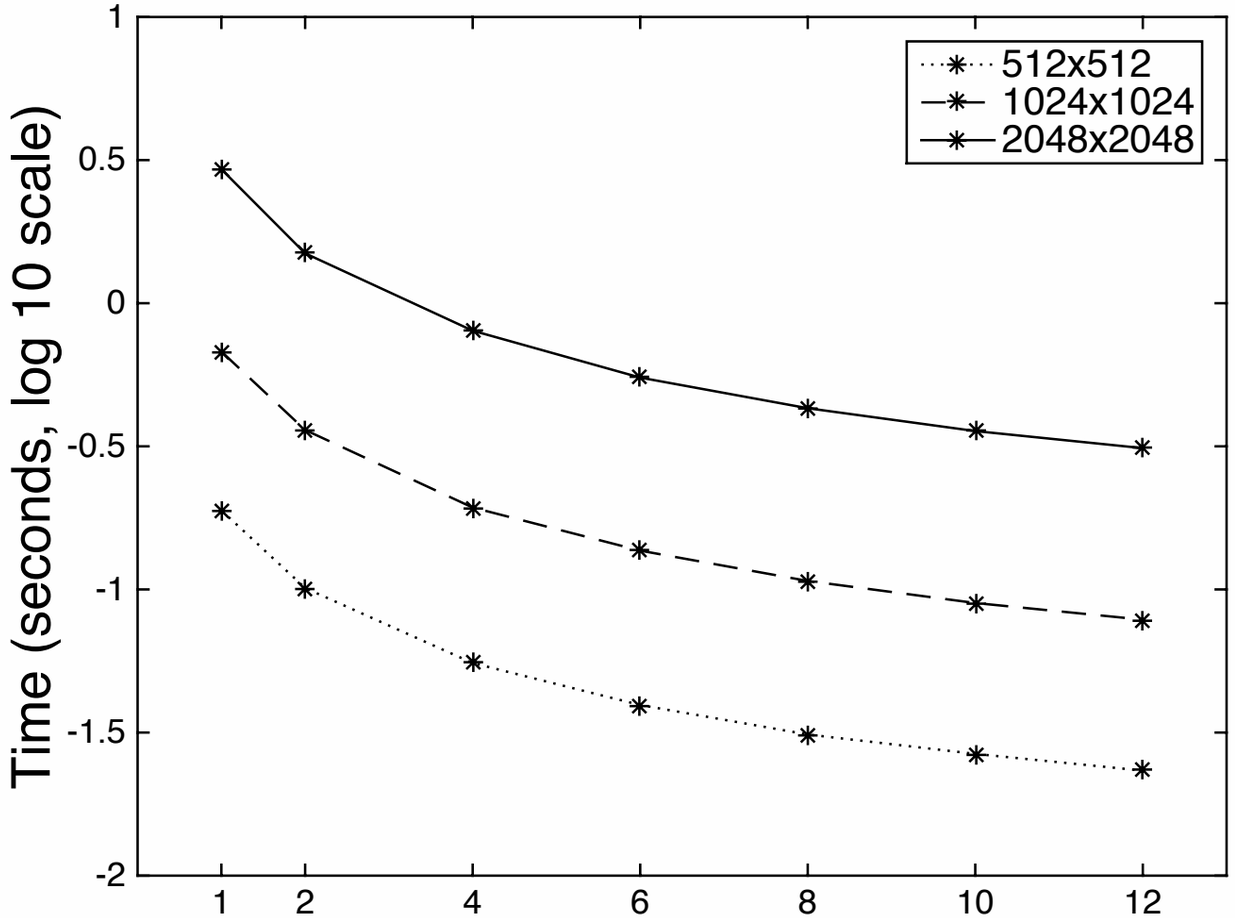
Fusion on parallel nodes and non parallel nodes

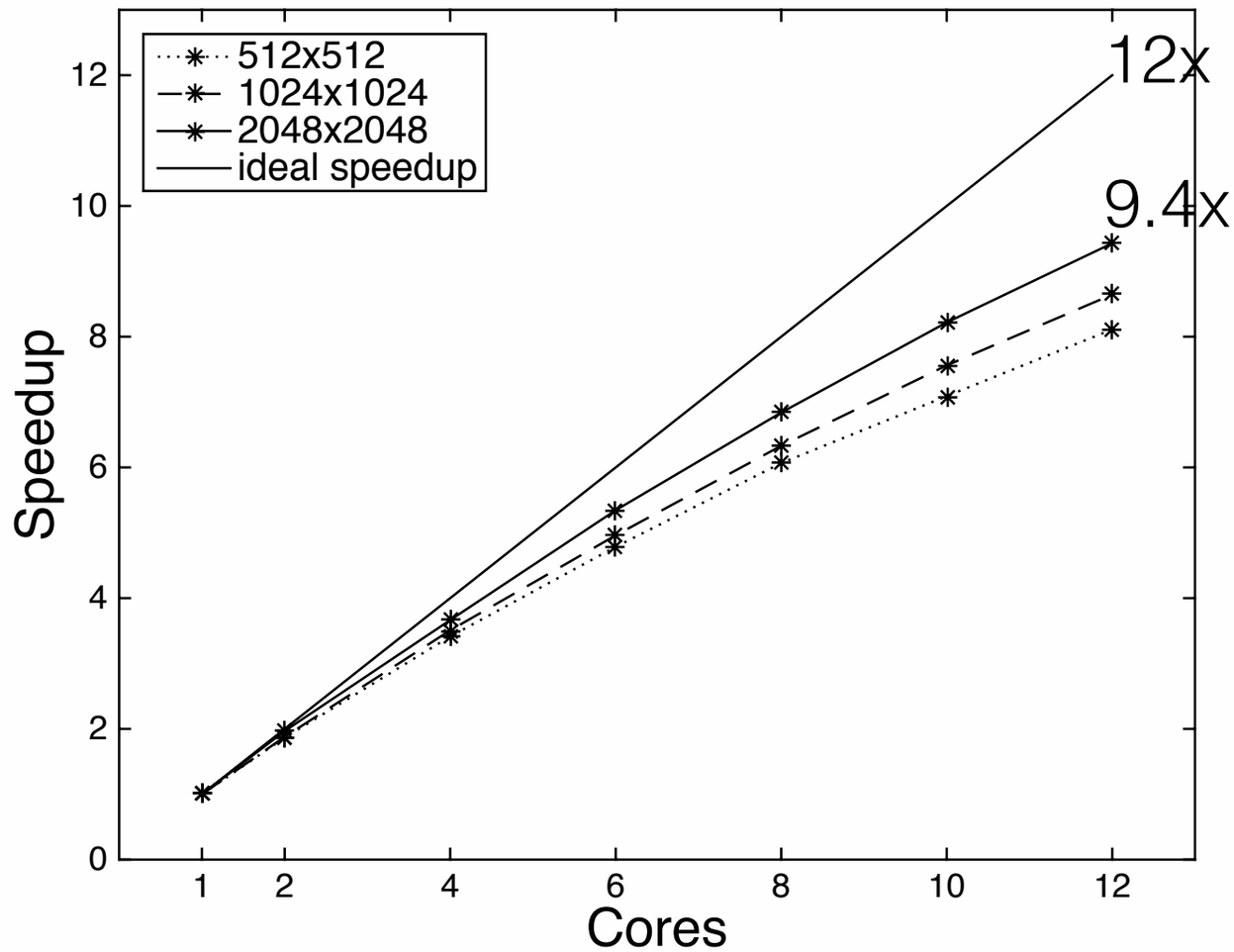
YAO applications

YAO is used in actual applications:

1. **Shallow-water**: 2D Shallow-water variational data assimilation
2. **Marine acoustics**: variational inversion of sound speed profile and retrieval of geoacoustic parameters (celerity, density, attenuation)
3. **Ocean colour**: variational inversion of multi-spectral satellite ocean colour measurements for the restitution of the chlorophyll-a
4. **PISCES**: ocean colour variational data assimilation in a biogeochemical model
5. **NEMO** (Nucleus for European Modelling of the Ocean) GYRE configuration: adjoint model
6. **SECHIBA**: ORCHIDEE land surface variational data assimilation

Performance results: Shallow-water



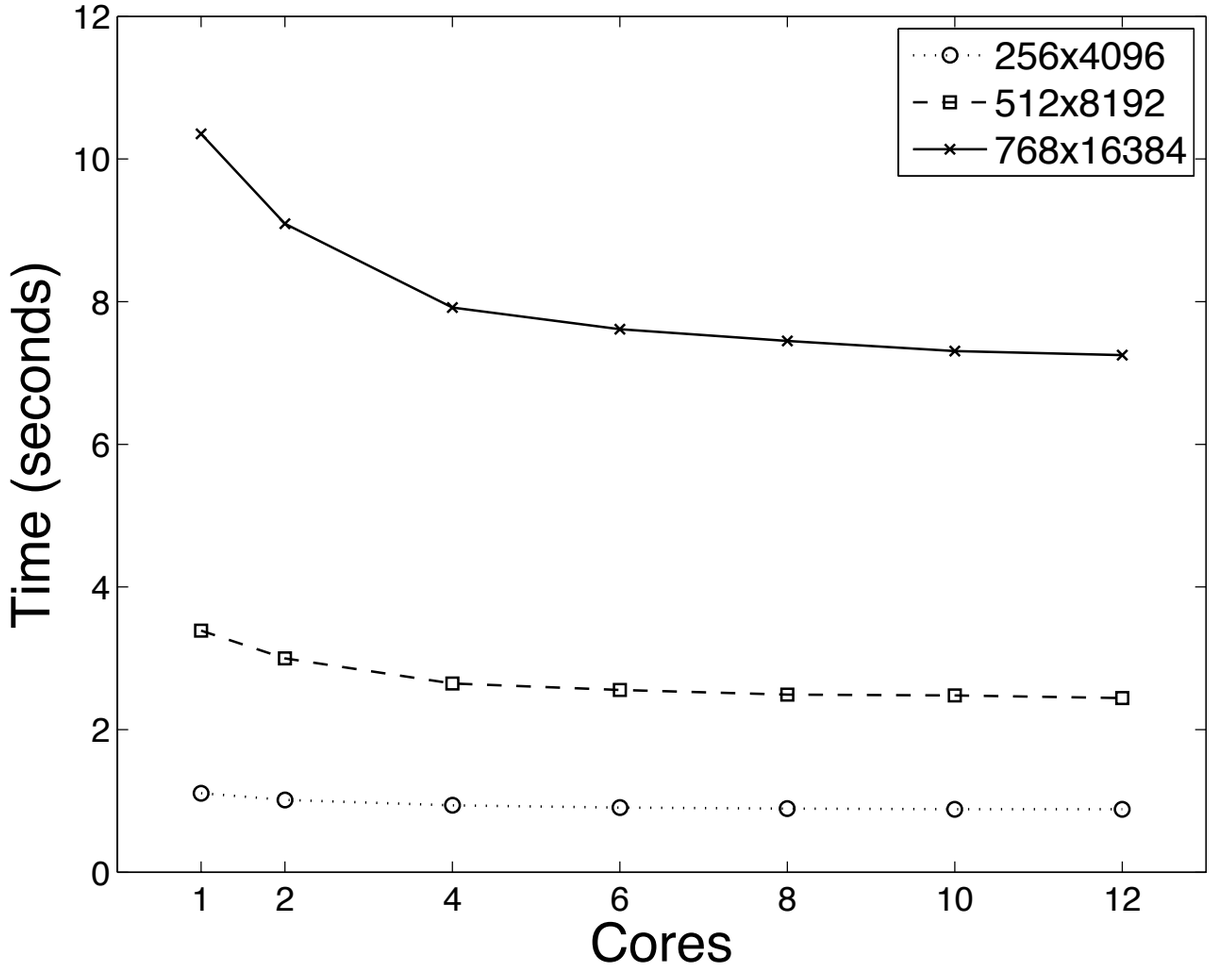


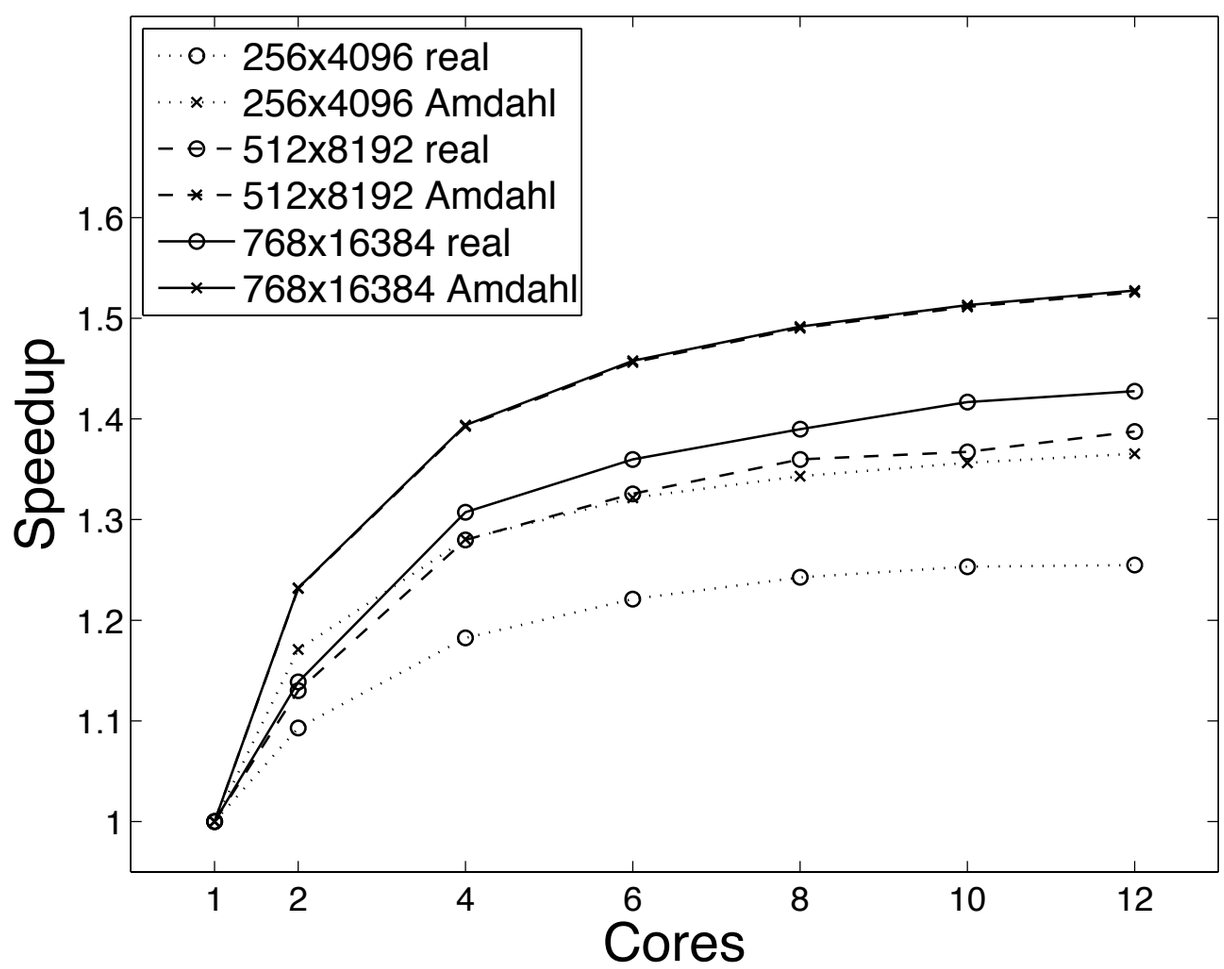
Runtime over one time step

6 modules, all automatically parallelised

AMD Magny-Cours Opteron 6168:
16 GB of memory,12 cores (1.9 GHz)

Performance results: marine acoustics



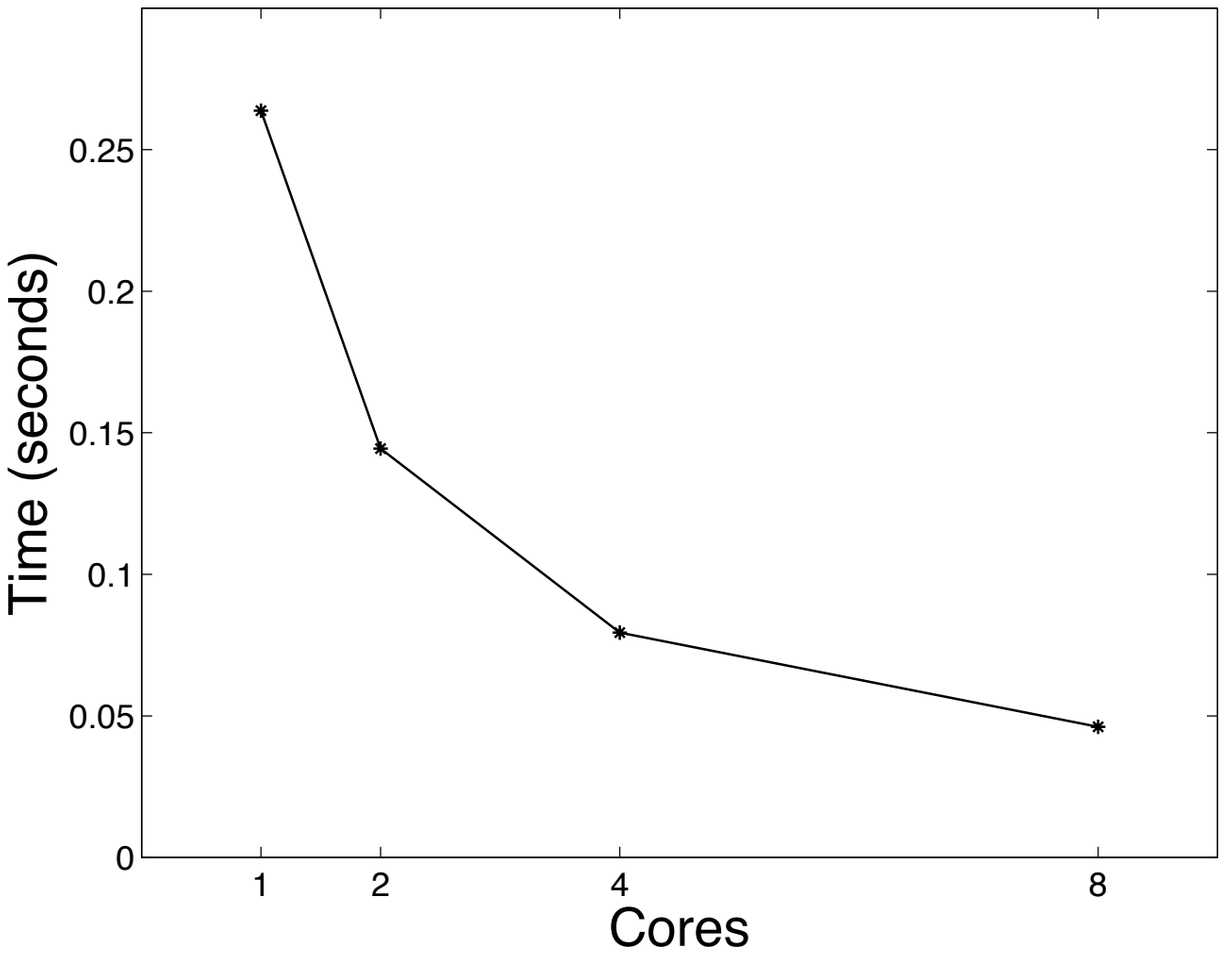


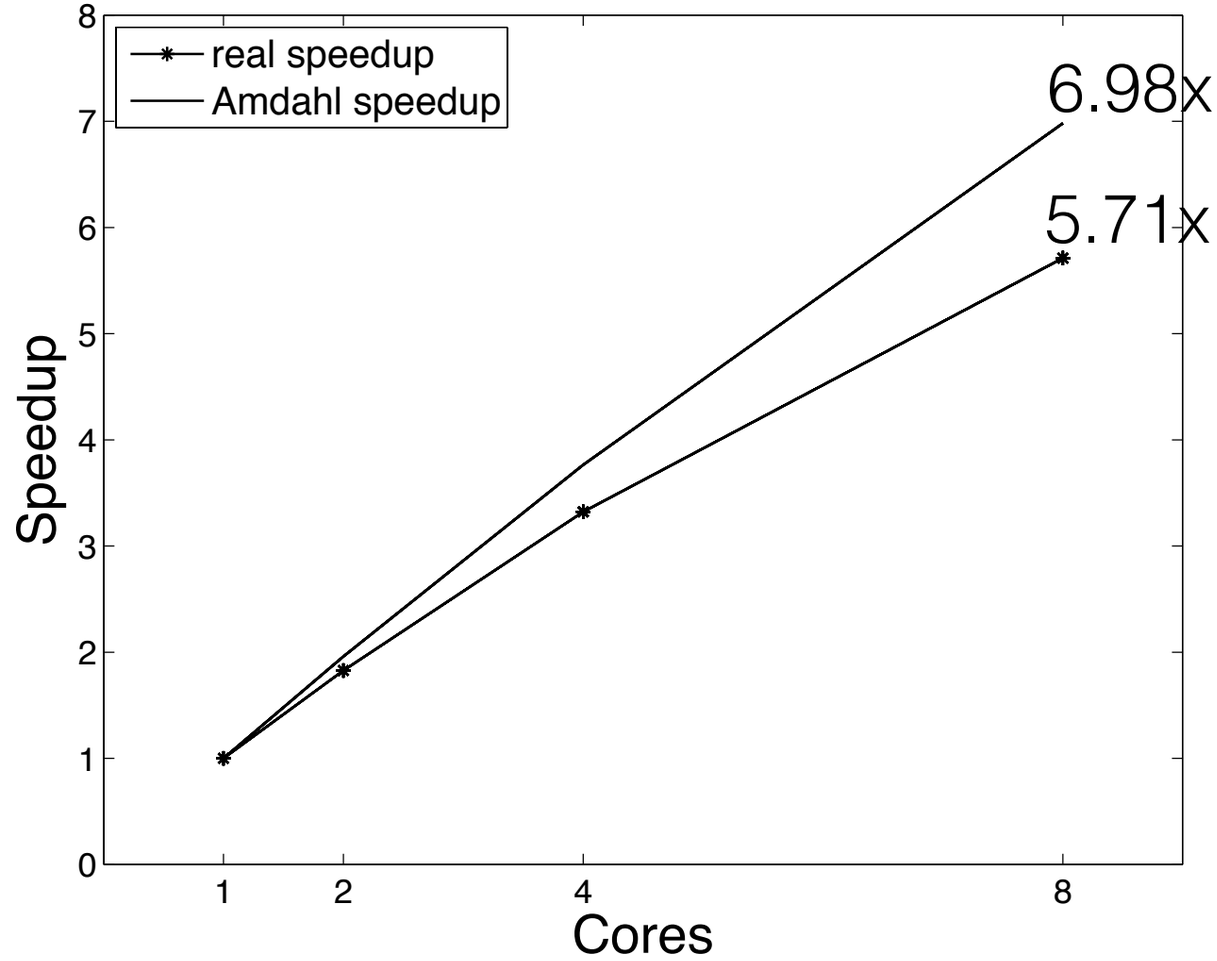
Runtime over one time step

9 modules, 6 automatically parallelised (most of the computation in the 3 non-parallel modules)

AMD Magny-Cours Opteron 6168:
16 GB of memory,12 cores (1.9 GHz)

Performance results: NEMO/GYRE





Runtime over one time step

GYRE: 32 × 22 × 31

82 modules, 80 automatically parallelised (2.1% is serial)

Speedup achieved 81.8% of theoretical TLP

AMD Magny-Cours Opteron 6168:
16 GB of memory,12 cores (1.9 GHz)

YAO DSL conclusions and future work

Conclusions

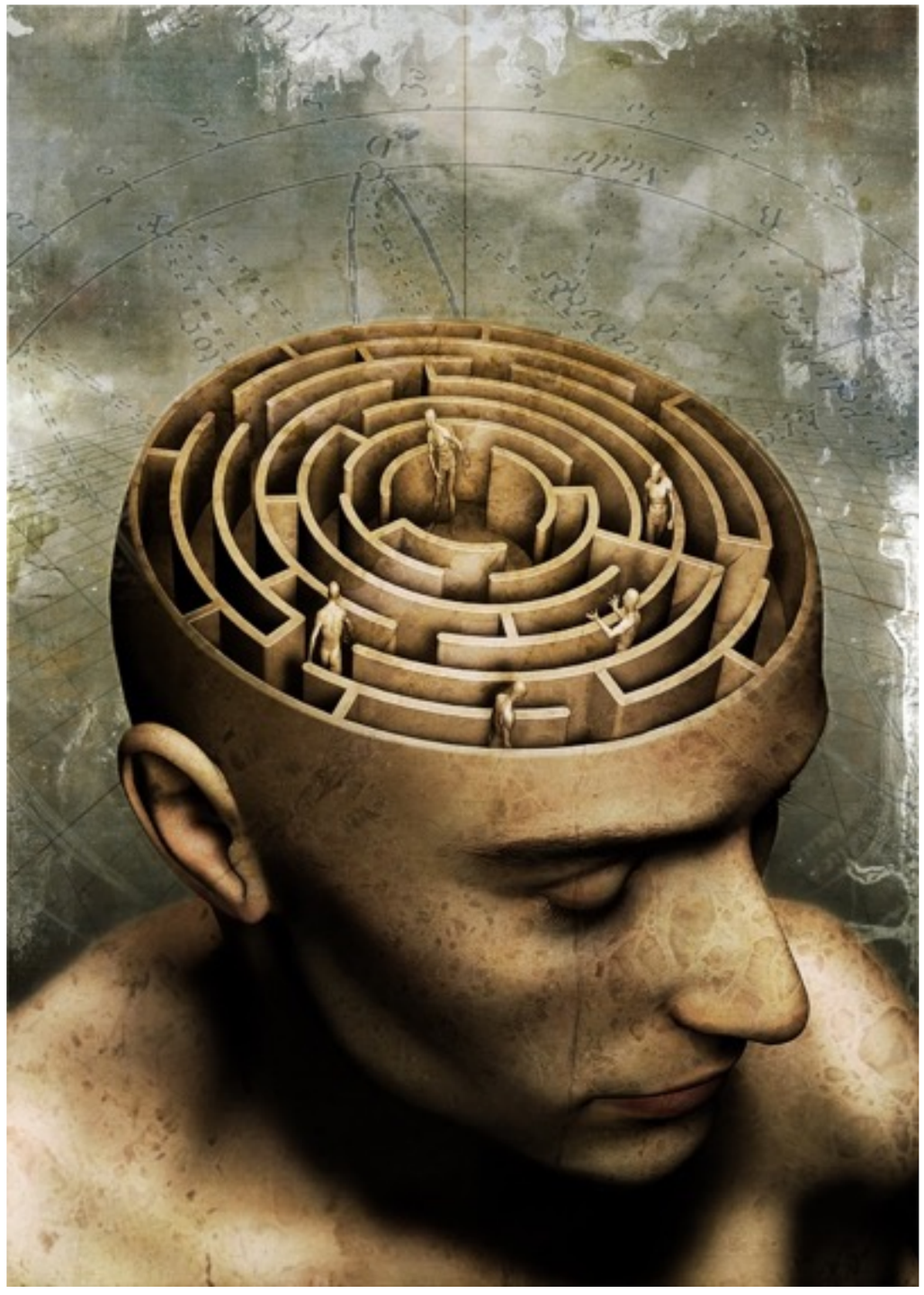
- A large community in geophysics exploits decades of research in automatic parallelisation without any additional effort and without any knowledge in parallel programming

Future work

- Direct application of the Polyhedral model: the code is affine
- Automatic generation of GPU code via OpenACC directives
- Automatic generation of MPI code
- Automatic checkpointing
- Automatic floating point numerical verification using CADNA

Outline

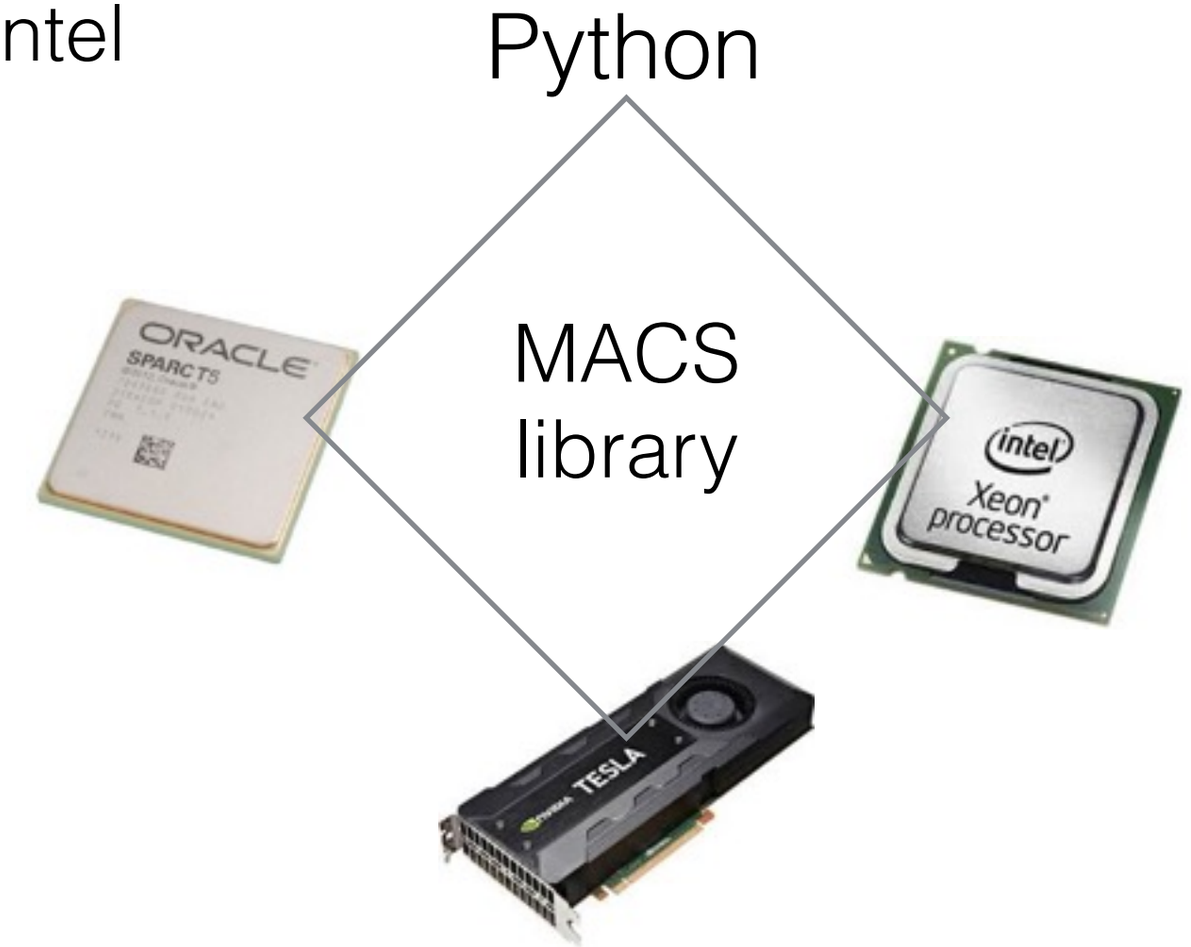
- Domain-specific languages, a practical use case
- **Micro-clusters of ARMs and GPUs**
- A holistic approach to performance and optimisation



MACS active library

[Spatz et al. 2010]

- Murex Analytics: MACS
- Real-world Monte Carlo and PDEs for computational finance
- Payoff language embedded in Python: active library
- Proprietary framework developed by Murex S.A.S.
- OpenCL kernels on NVIDIA and Intel
- C++ on Intel and SPARC



MPMD CPU/GPU cluster architecture

Client-server design pattern

Specialised for sequential code

Py + C + MPI

Fast x86 node

MPI

C + MPI (IB) + OpenCL

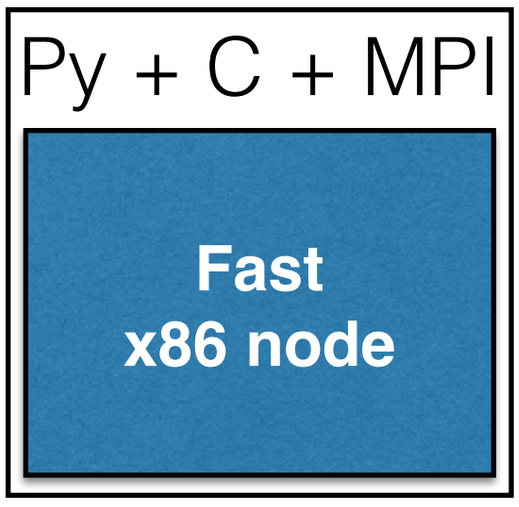
Node 1 2 x86 2 M2090	**Node 2 2 x86 2 M2090**	**Node 3 2 x86 2 M2090**
Node 4 2 x86 2 M2090	**Node 5 2 x86 2 M2090**	**Node 6 2 x86 2 M2090**
Node 7 2 x86 2 M2090	**Node 8 2 x86 2 M2090**	**Node 9 2 x86 2 M2090**

- Extending the OpenCL standard to work on a cluster of GPUs
- Linear speedup: non-blocking MPI communications, PRNG, Monte Carlo

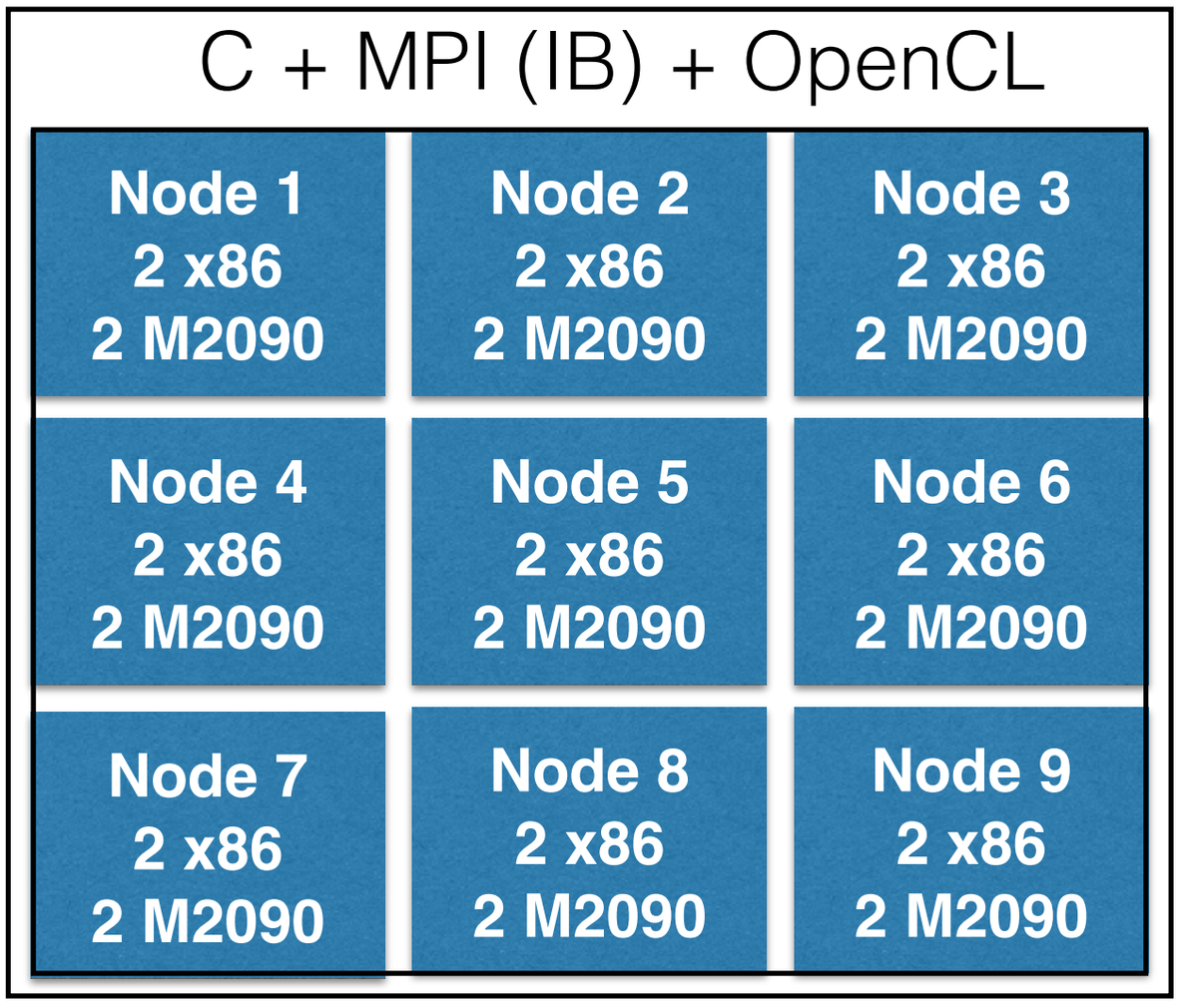
MPMD CPU/GPU cluster architecture

Client-server design pattern

Specialised for sequential code



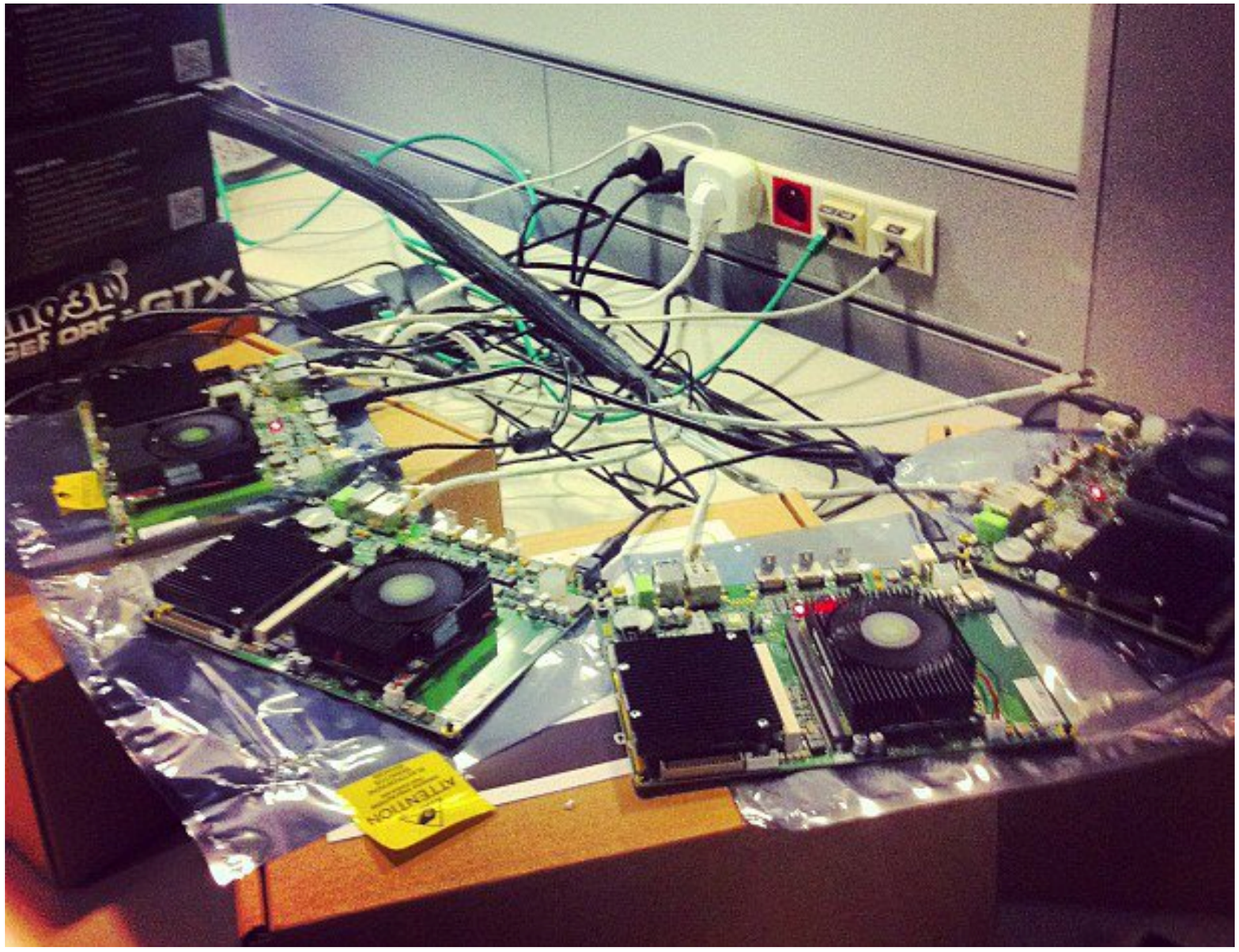
MPI



CPU code is an OCL wrapper, RAM and HD not exploited!

- Extending the OpenCL standard to work on a cluster of GPUs
- Linear speedup: non-blocking MPI communications, PRNG, Monte Carlo

CARMA (CUDA on ARM) micro-cluster



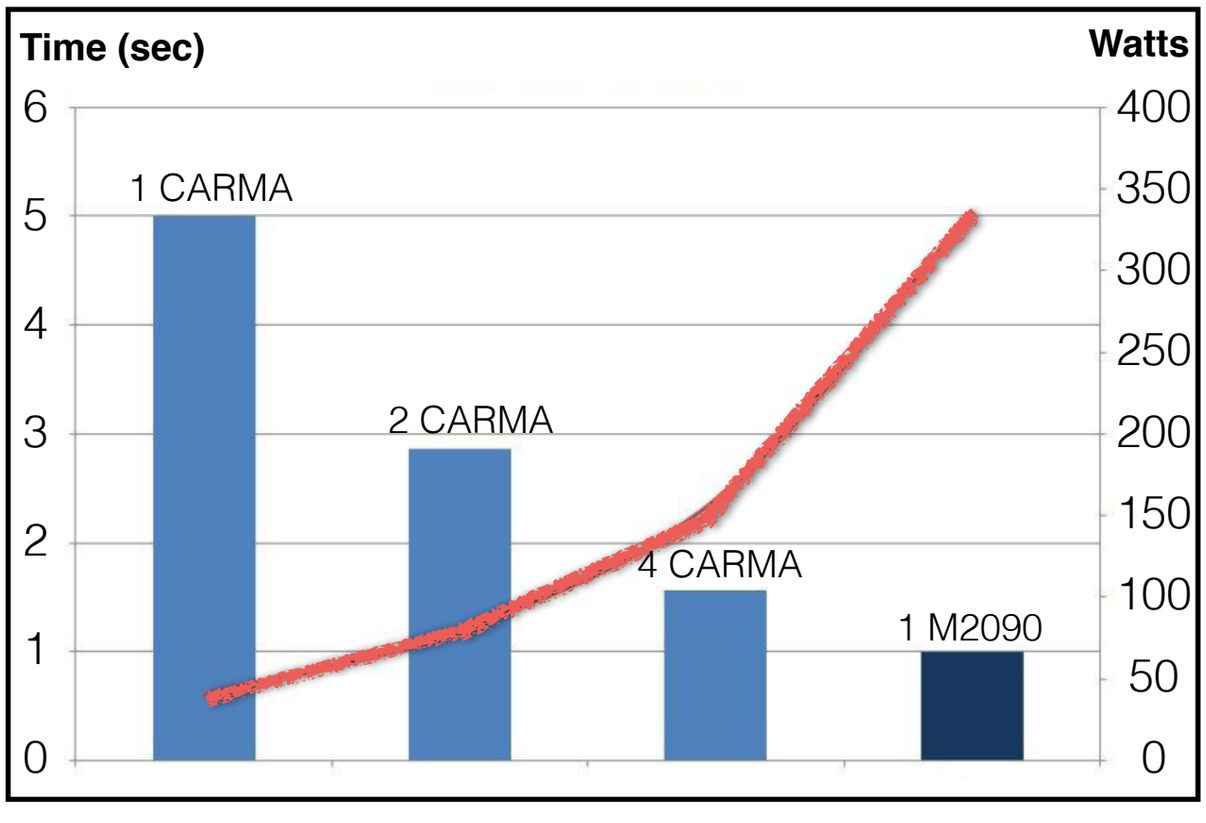
Micro-cluster a real test case

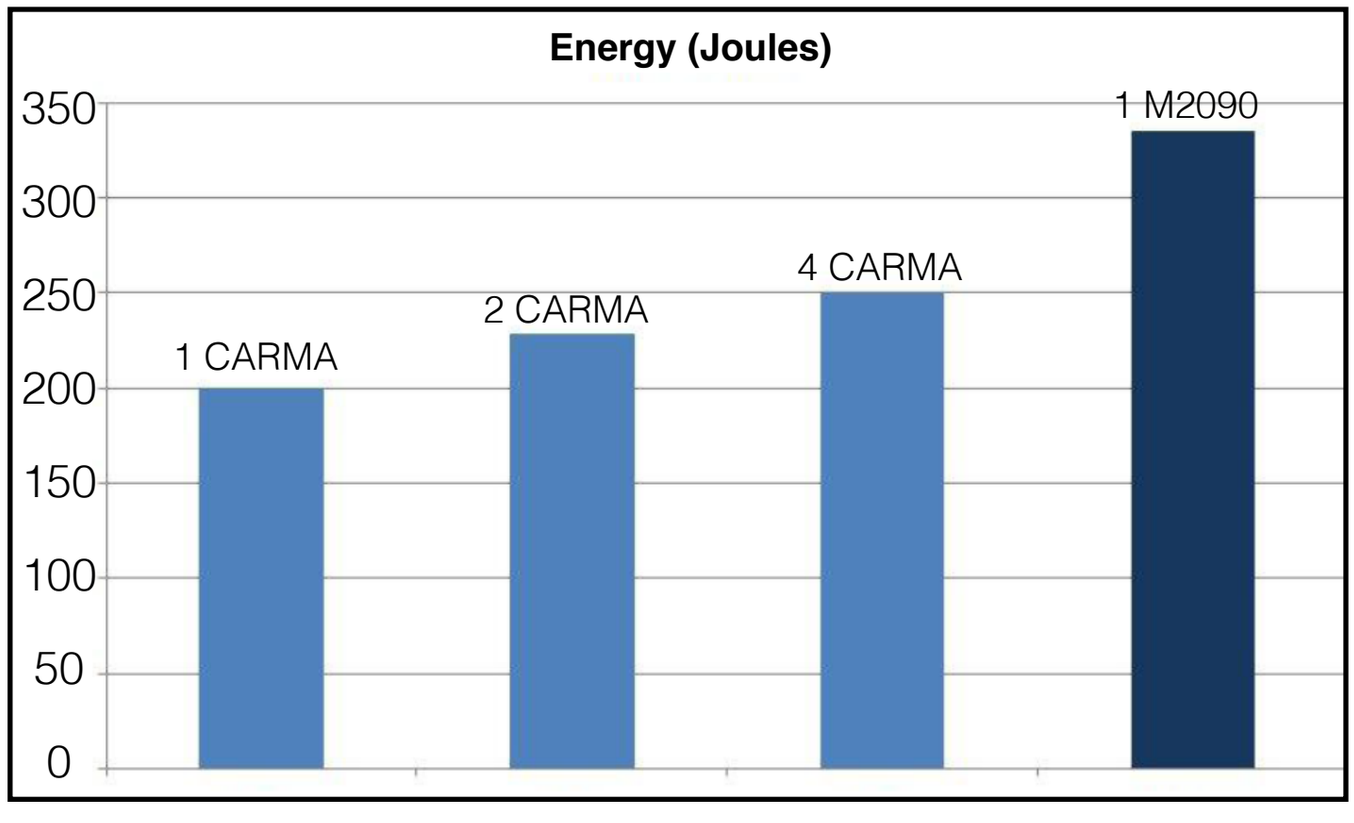
	x86 + x86 host + M2090	**x86 + 4 CARMAs**	
Mem (GB)	6	8	
Peak SP GFLOPS	1331	1080	ratio=0.8
D2D BW (GB/s)	138	96	ratio=0.7
H2D (GB/s)	2.2	0.3	ratio=0.1
GPU power (W)	225 theoretical	160 measured	
x86 host power (W)	100	0	
Max power (W)	325	180	ratio=2
Network	IB	1 Gb Ethernet	

Micro-cluster a real test case

	x86 + x86 host + M2090	**x86 + 4 CARMAs**	
Mem (GB)	6	8	
Peak SP GFLOPS	1331	1080	ratio=0.8
D2D BW (GB/s)	138	96	ratio=0.7
H2D (GB/s)	2.2	0.3	ratio=0.1
GPU power (W)	225 theoretical	160 measured	
x86 host power (W)	100	0	
Max power (W)	325	180	ratio=2
Network	IB	1 Gb Ethernet	

Application: 365 days Asian option single stock with BS MC: PRNG, Brownian bridge, BS diffusion, payoff





Conclusions CARMA MC benchmarking

1. Runtime and energy consumption:
   - CARMAs runtime close to Tesla M2090 runtime
   - CARMAs more efficient in terms of energy

Conclusions CARMA MC benchmarking

1. Runtime and energy consumption:
   - CARMAs runtime close to Tesla M2090 runtime
   - CARMAs more efficient in terms of energy
2. MACS DSL users automatically exploit the cluster computation

Conclusions CARMA MC benchmarking

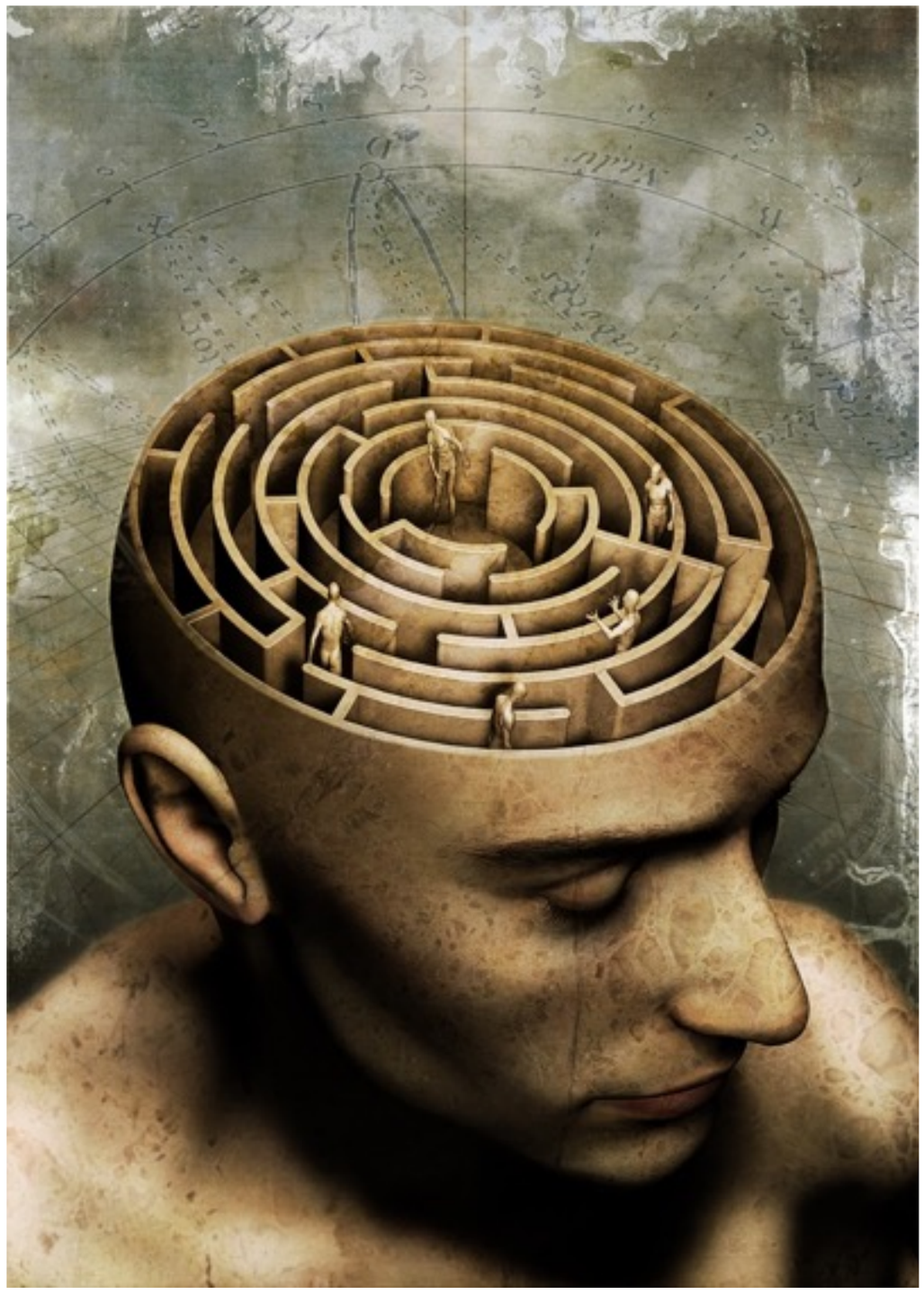
1. Runtime and energy consumption:
   - CARMAs runtime close to Tesla M2090 runtime
   - CARMAs more efficient in terms of energy
2. MACS DSL users automatically exploit the cluster computation
3. CARMA devkit (2012) inspired next generation SoC boards

Conclusions CARMA MC benchmarking

1. Runtime and energy consumption:
   - CARMAs runtime close to Tesla M2090 runtime
   - CARMAs more efficient in terms of energy
2. MACS DSL users automatically exploit the cluster computation
3. CARMA devkit (2012) inspired next generation SoC boards
4. Widely used in robotics, will it reach the supercomputing segment for exascale?
   - Mont Blanc, Denver projects

Outline

- Domain-specific languages, a practical use case
- Micro-clusters of ARMs and GPUs
- **A holistic approach to performance and optimisation**



What is “Performance”?

1. In several domains performance is **execution time**
2. In some domains performance is **accuracy**
3. What about **energy**?
4. But also memory consumption, temperature, robustness, etc.

A modern system evaluation considers the 3 metrics or more:

$$Performance = \begin{bmatrix} Runtime \\ Energy \\ Accuracy \end{bmatrix}$$

This defines a multi-objective optimisation problem

“Performance” on SLAMBench

SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

[Nardi et al., 2015]

“Performance” on SLAMBench

SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

Machine	CPU	CPU name	CPU GFLOPS	CPU cores	GPU	GPU name	GPU GFLOPS	TDP Watts
Hardkernel ODROID-XU3	ARM A15 + A7	Exynos 5422	80	4 + 4	ARM	Mali-T628	60 + 30	10

[Nardi et al., 2015]

“Performance” on SLAMBench

SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

ATE in cm	
C++	2.06
OpenMP	2.06
OpenCL	2.01

Machine	CPU	CPU name	CPU GFLOPS	CPU cores	GPU	GPU name	GPU GFLOPS	TDP Watts
Hardkernel ODROID-XU3	ARM A15 + A7	Exynos 5422	80	4 + 4	ARM	Mali-T628	60 + 30	10

[Nardi et al., 2015]

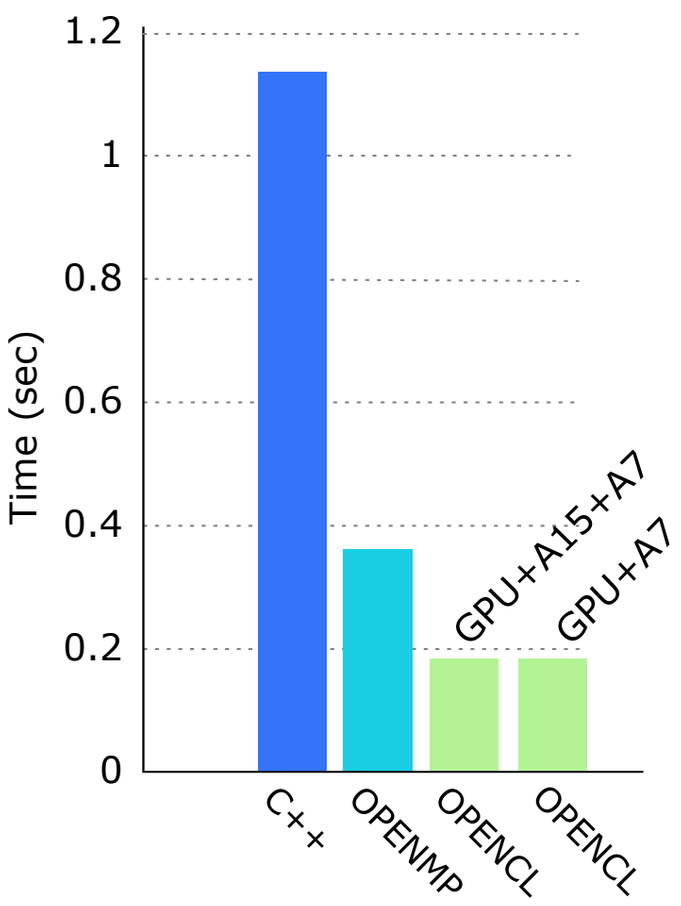
“Performance” on SLAMBench

SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

ATE in cm	
C++	2.06
OpenMP	2.06
OpenCL	2.01

Machine	CPU	CPU name	CPU GFLOPS	CPU cores	GPU	GPU name	GPU GFLOPS	TDP Watts
Hardkernel ODROID-XU3	ARM A15 + A7	Exynos 5422	80	4 + 4	ARM	Mali-T628	60 + 30	10



[Nardi et al., 2015]

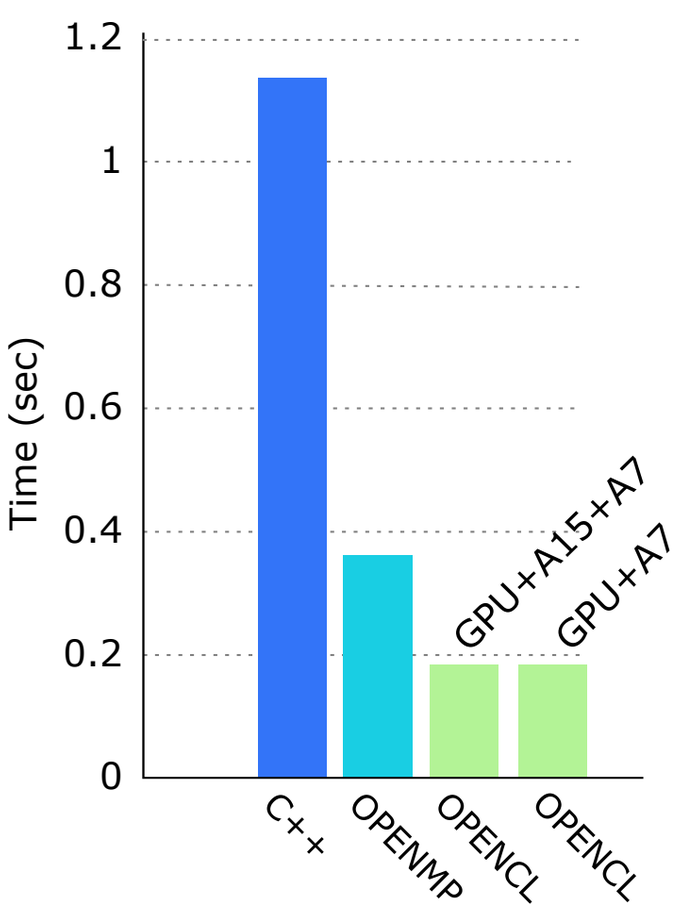
“Performance” on SLAMBench

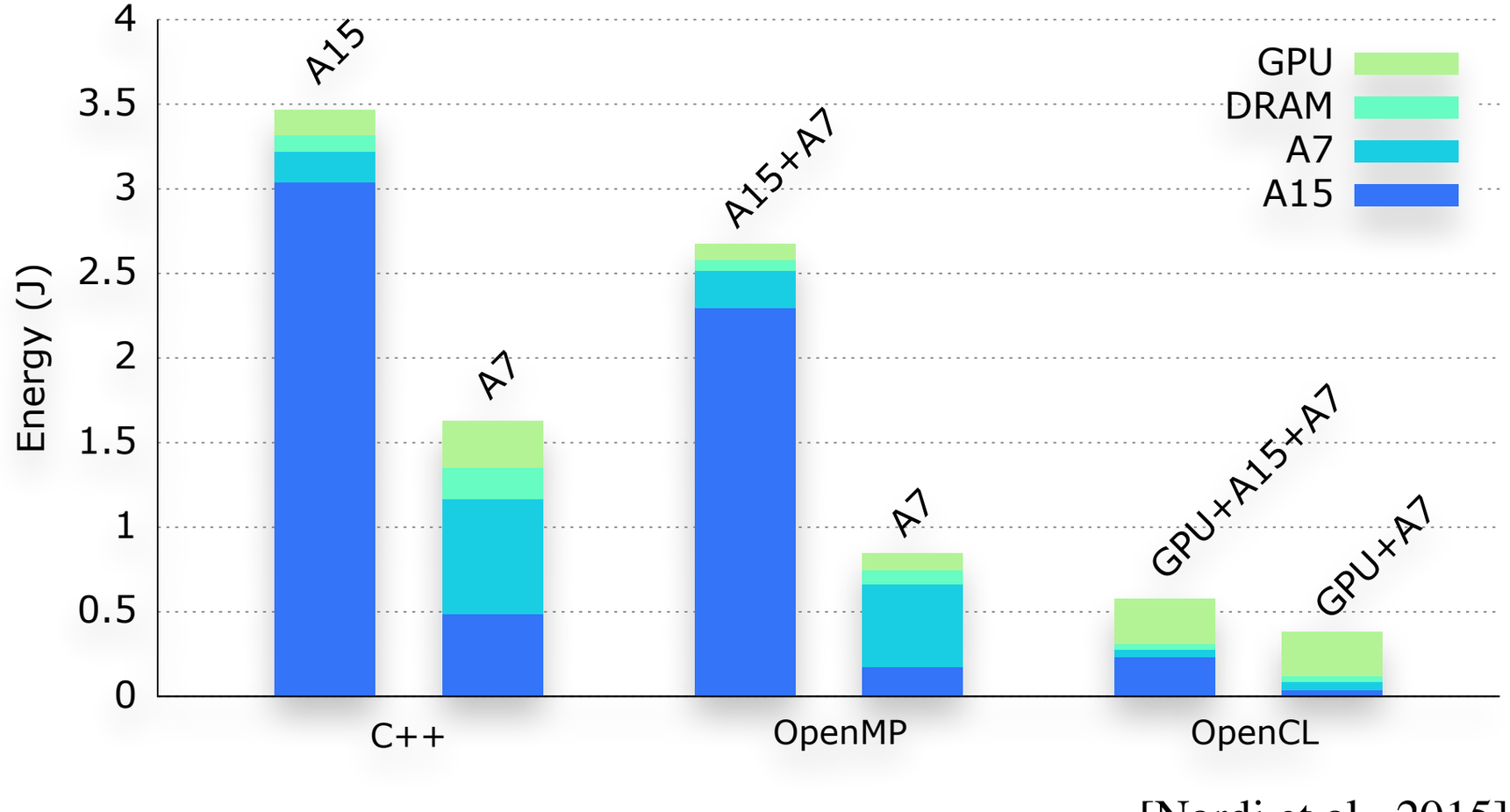
SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

ATE in cm	
C++	2.06
OpenMP	2.06
OpenCL	2.01

Machine	CPU	CPU name	CPU GFLOPS	CPU cores	GPU	GPU name	GPU GFLOPS	TDP Watts
Hardkernel ODROID-XU3	ARM A15 + A7	Exynos 5422	80	4 + 4	ARM	Mali-T628	60 + 30	10





[Nardi et al., 2015]

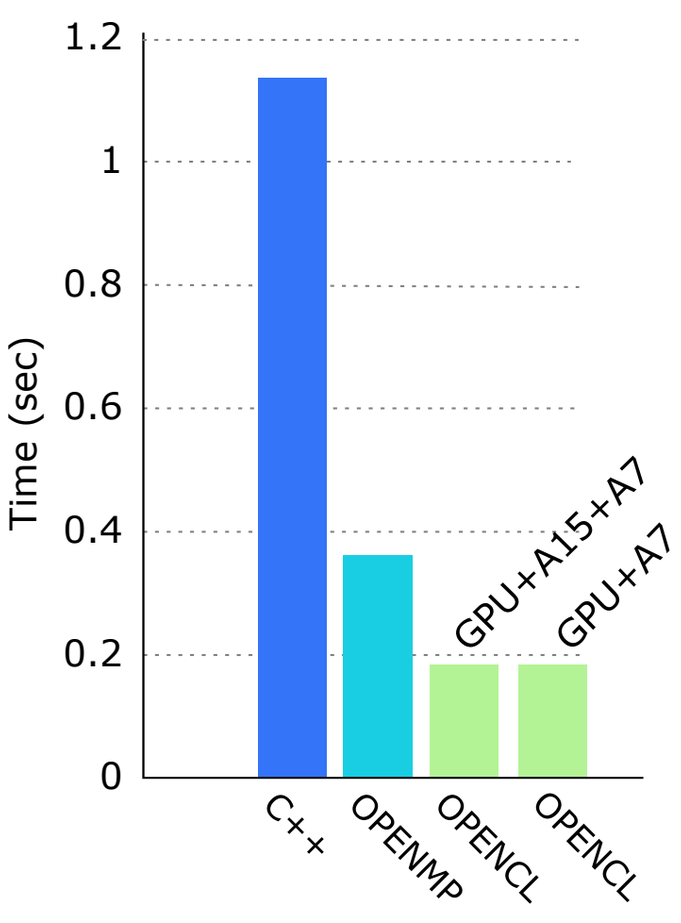
“Performance” on SLAMBench

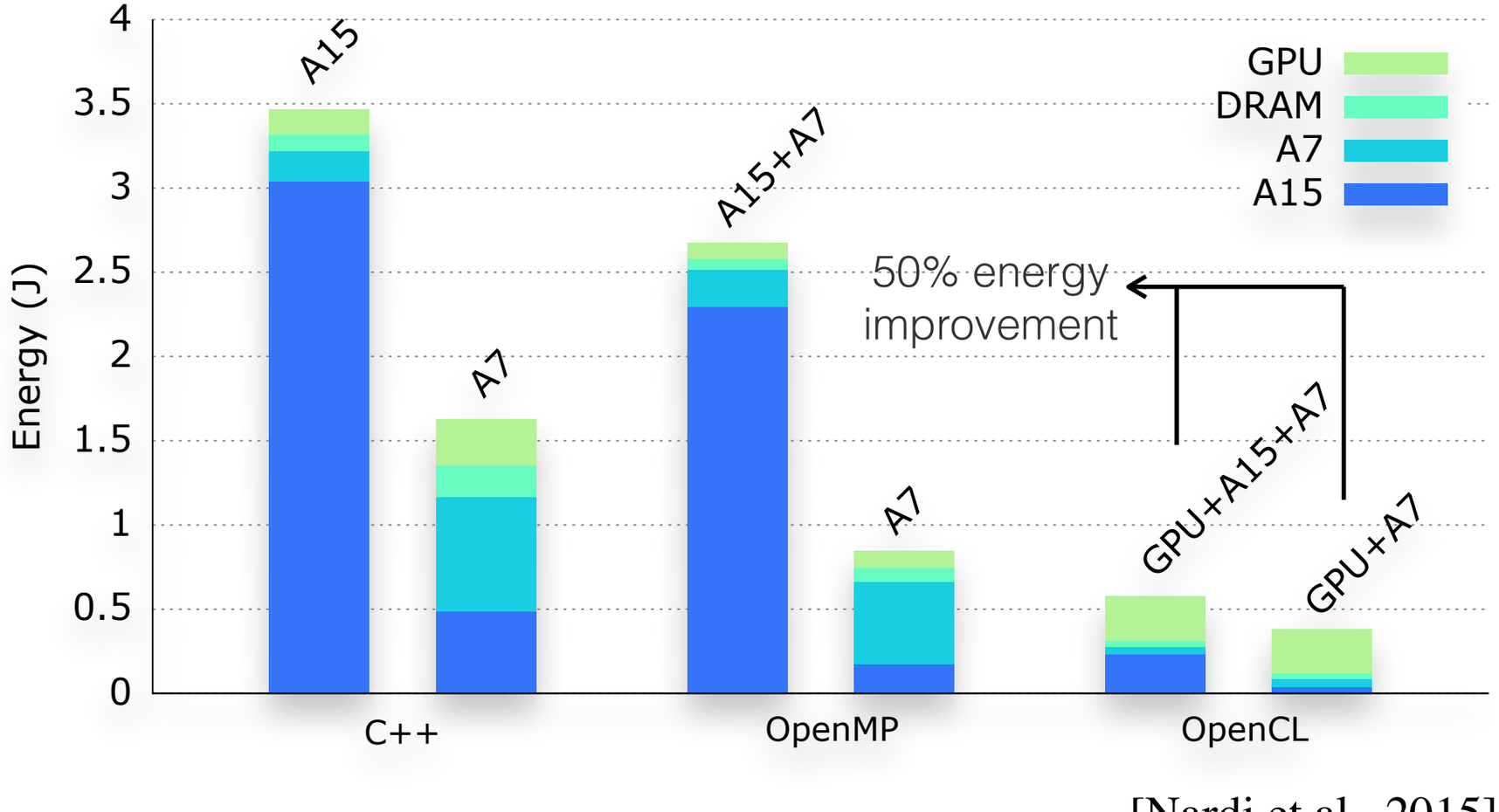
SLAMBench is a benchmark for 3D scene understanding:

- Provides runtime/energy/accuracy measurements
- Provides accuracy via absolute trajectory error (ATE)
- Designed for holistic approach to “performance”

ATE in cm	
C++	2.06
OpenMP	2.06
OpenCL	2.01

Machine	CPU	CPU name	CPU GFLOPS	CPU cores	GPU	GPU name	GPU GFLOPS	TDP Watts
Hardkernel ODROID-XU3	ARM A15 + A7	Exynos 5422	80	4 + 4	ARM	Mali-T628	60 + 30	10





[Nardi et al., 2015]

What is the optimisation space?

Configuration parameters:

Space 1

1. Algorithmic:
   - Minimisation methods
   - Early exit condition value
   - Application-specific parameters

What is the optimisation space?

Configuration parameters:

Space 1

1. Algorithmic:
   - Minimisation methods
   - Early exit condition value
   - Application-specific parameters

Space 2

2. Compilation:
   - opencl-params: -cl-mad-enable,-cl-fast-relaxed-math, etc.
   - Local work group size
   - Kernel partitioning ratio
   - Thread coarsening

What is the optimisation space?

Configuration parameters:

Space 1

1. Algorithmic:
   - Minimisation methods
   - Early exit condition value
   - Application-specific parameters

Space 2

2. Compilation:
   - opencl-params: -cl-mad-enable,-cl-fast-relaxed-math, etc.
   - Local work group size
   - Kernel partitioning ratio
   - Thread coarsening

Space 3

3. Hardware:
   - CPU frequency
   - # of active big cores
   - # of active LITTLE cores

What is the optimisation space?

Configuration parameters:

Joint space

Space 1

1. Algorithmic:
   - Minimisation methods
   - Early exit condition value
   - Application-specific parameters

Space 2

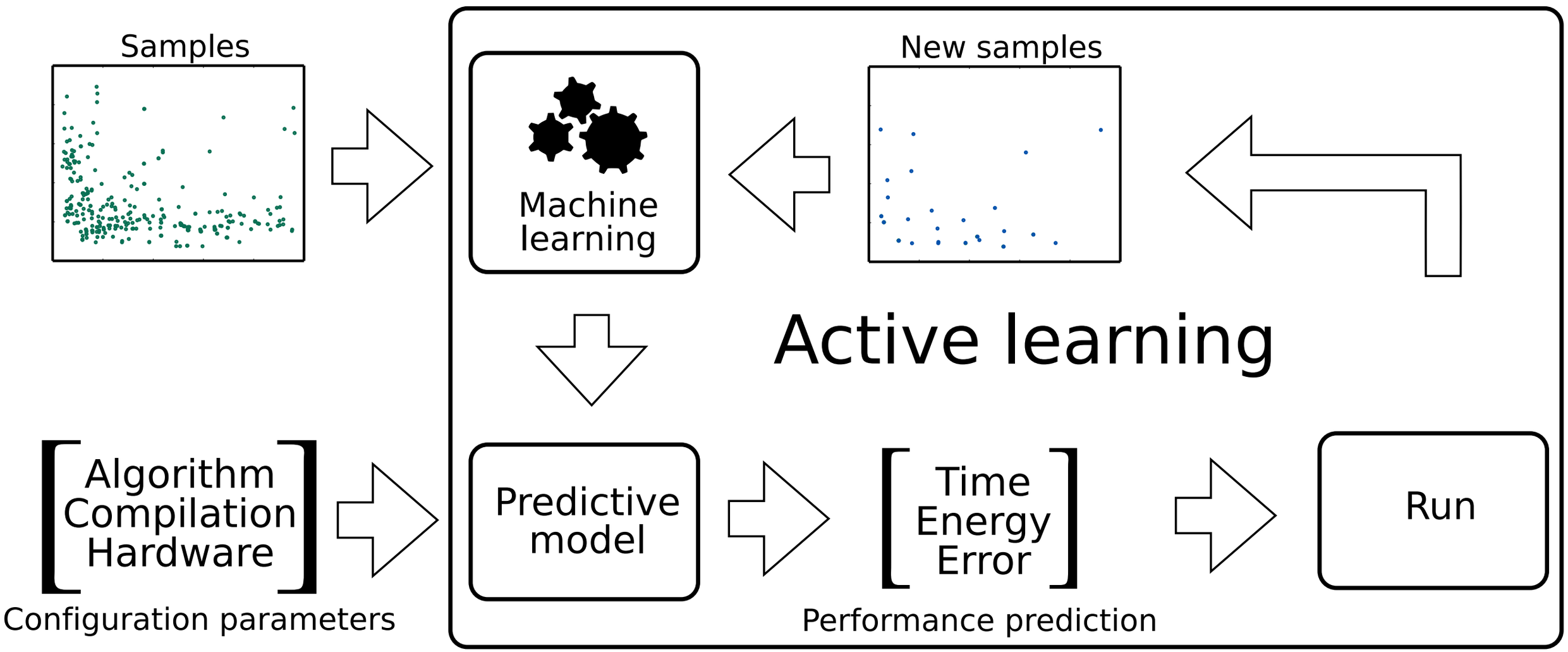
2. Compilation:
   - opencl-params: -cl-mad-enable,-cl-fast-relaxed-math, etc.
   - Local work group size
   - Kernel partitioning ratio
   - Thread coarsening

Space 3

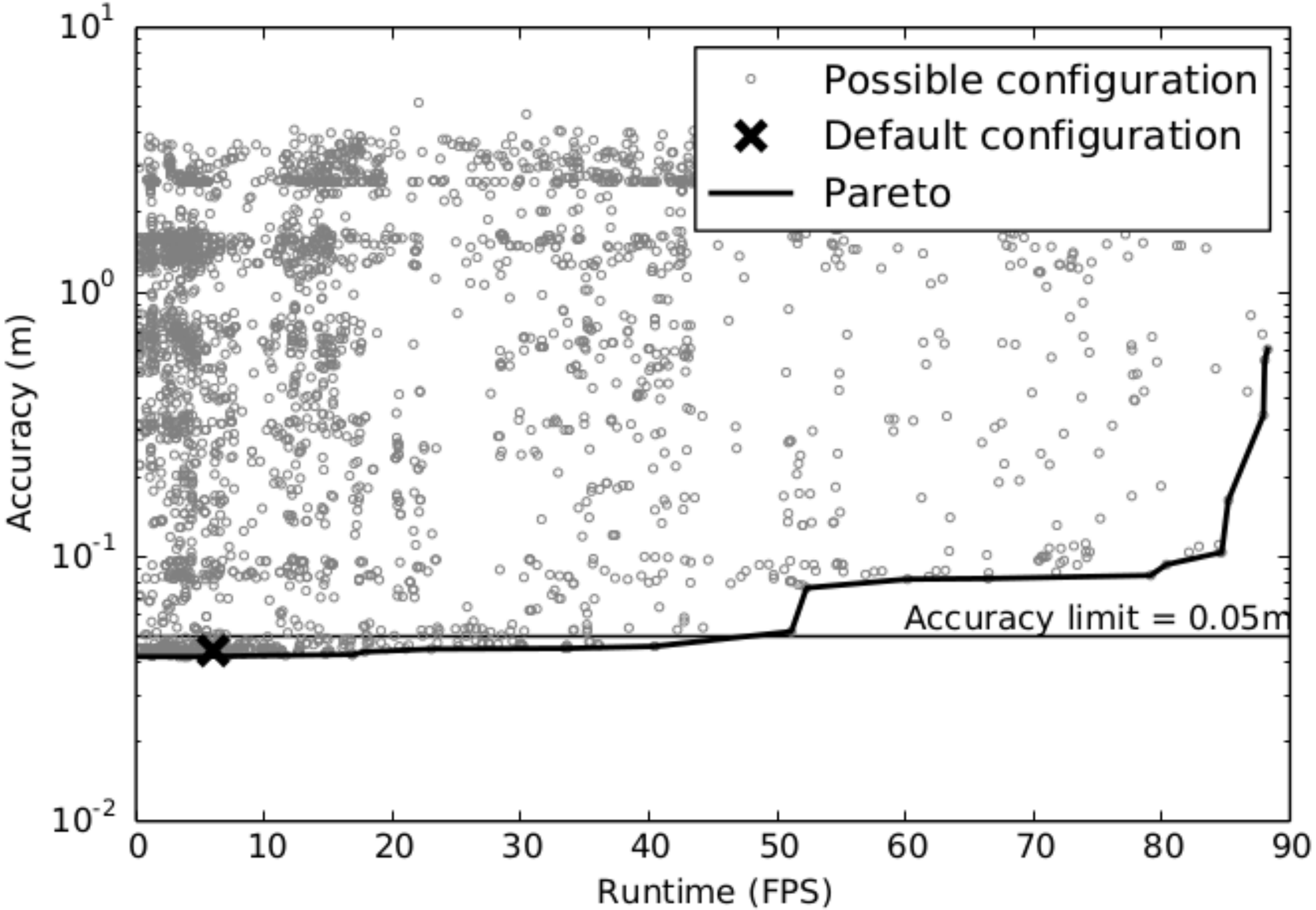
3. Hardware:
   - CPU frequency
   - # of active big cores
   - # of active LITTLE cores

Warning: huge space, impossible to run exhaustively

Joint design-space exploration (DSE)



DSE on SLAMBench algorithmic parameters

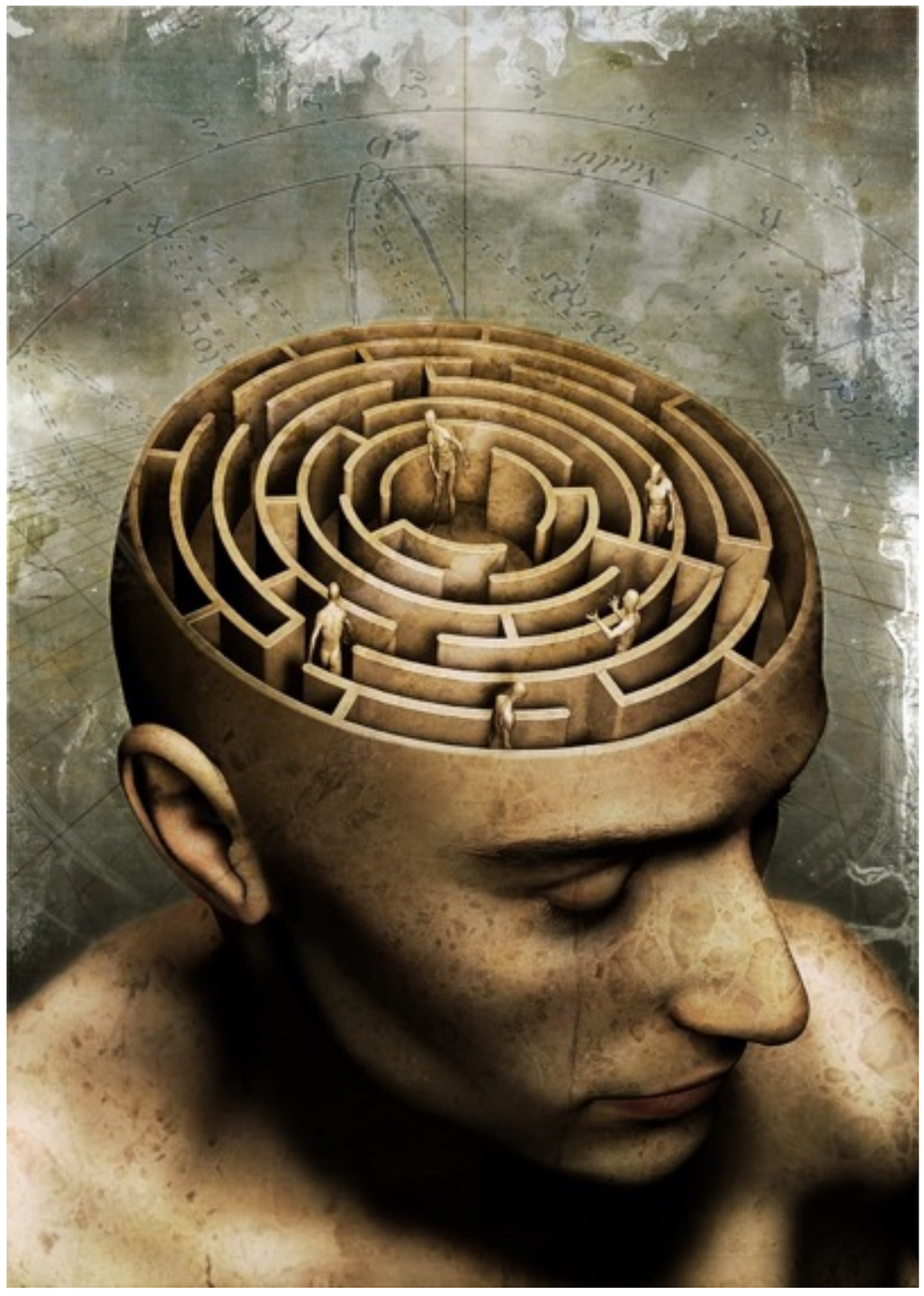


Multi-objective DSE conclusions

- Multi-objective “performance” optimisation: runtime/energy/accuracy
- Joint design-space highlights exploration opportunities
- Autotuning tools to pick interesting points
- This enables autotuning at the DSL level

Outline

- Domain-specific languages, a practical use case
- Micro-clusters of ARMs and GPUs
- A holistic approach to performance and optimisation

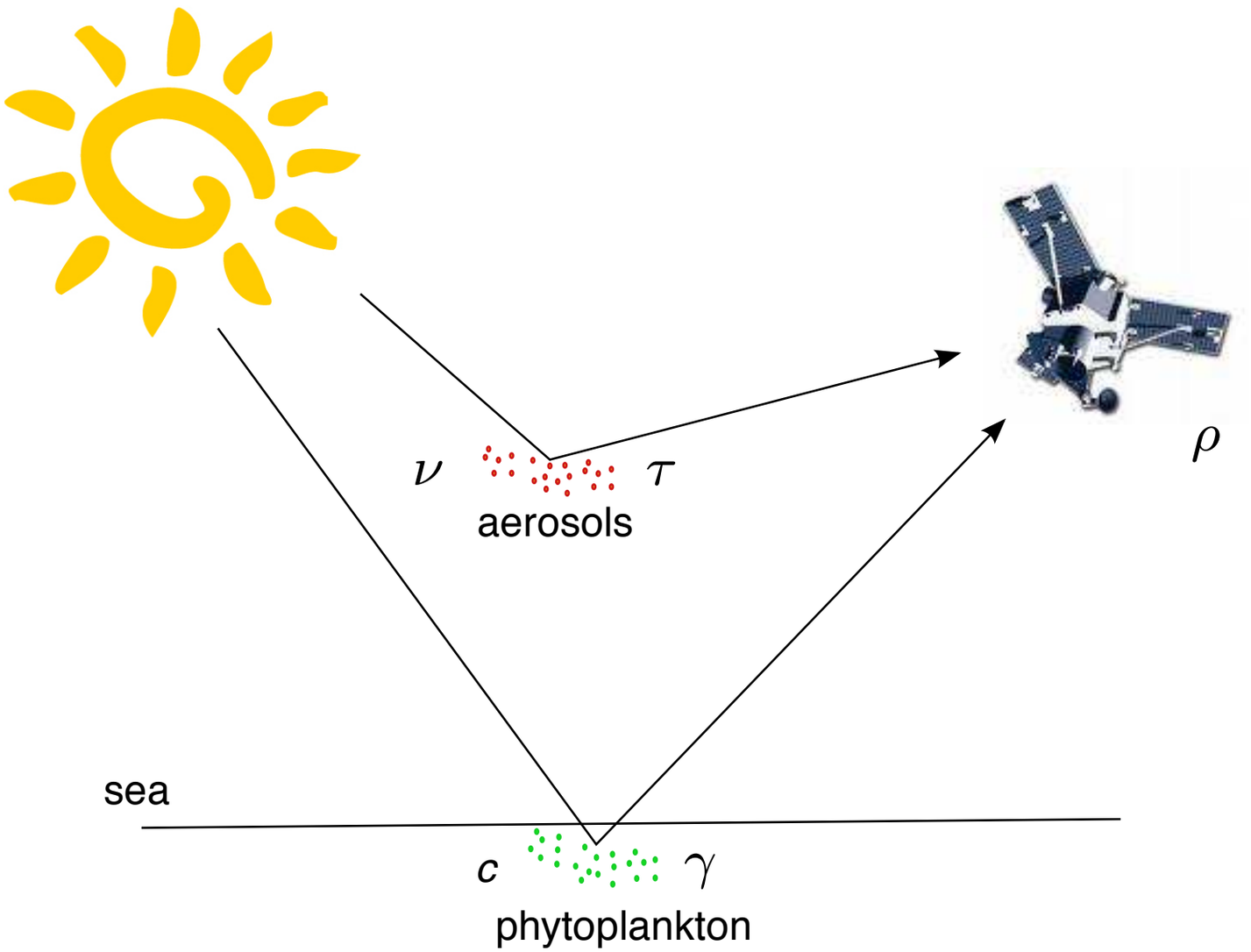


References

1. [Nardi et al., 2015] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Luján, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber. "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM." Submitted, arXiv:1410.2167, 2015.
2. [Newcombe et al., 2011] R. A. Newcombe, S. J. Lovegrove and A. J. Davison. "DTAM: Dense tracking and mapping in real-time." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.
3. [Spatz et al., 2010] P. Spatz, "Practical Methods Beyond Monte Carlo in Finance." GPU Technology Conference (GTC), 2010.
4. [Nardi et al., 2009] L. Nardi, C. Sorror, F. Badra, and S. Thiria. "YAO: a software for variational data assimilation using numerical models." Computational Science and Its Applications–ICCSA 2009.
5. [Nardi et al., 2012] L. Nardi, F. Badran, P. Fortin and S. Thiria. "YAO: a generator of parallel code for variational data assimilation applications." In IEEE High Performance Computing and Communication HPCC, 2012.
6. Berrada, Mohamed. "Une approche variationnelle de l'inversion, de la recherche localea la recherche globale par carte topologique: application en inversion géoacoustique." PhD Thesis, UPMC, France, 2008.

Backup slides

Data assimilation example



- τ: aerosols concentratio
- ν: aerosols size
- C: phytoplankton concentration
- γ: phytoplankton size
- ρ:

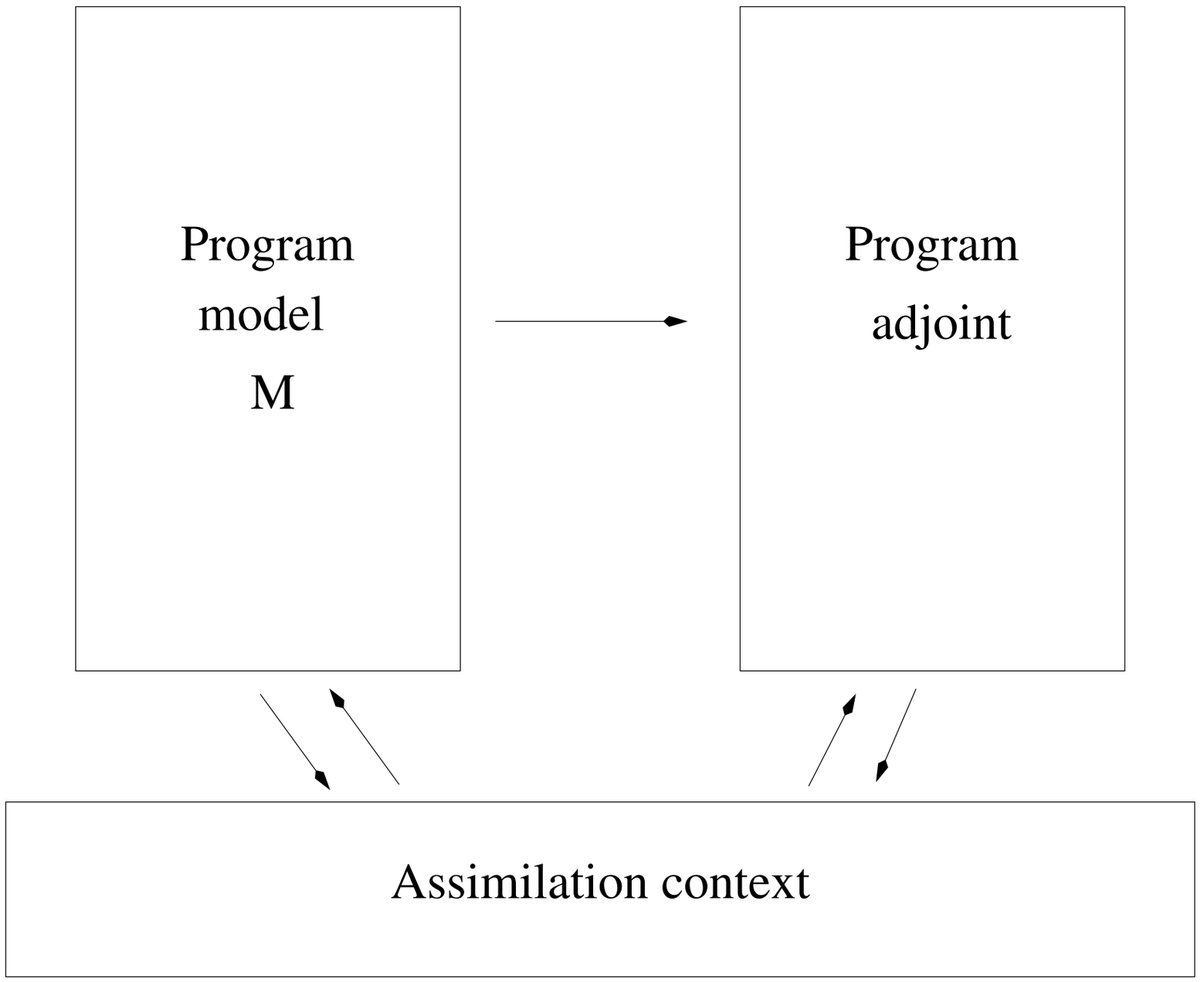
$$\rho = \boldsymbol{M}(\tau, \nu, \gamma, \boldsymbol{c})$$

YAO front end

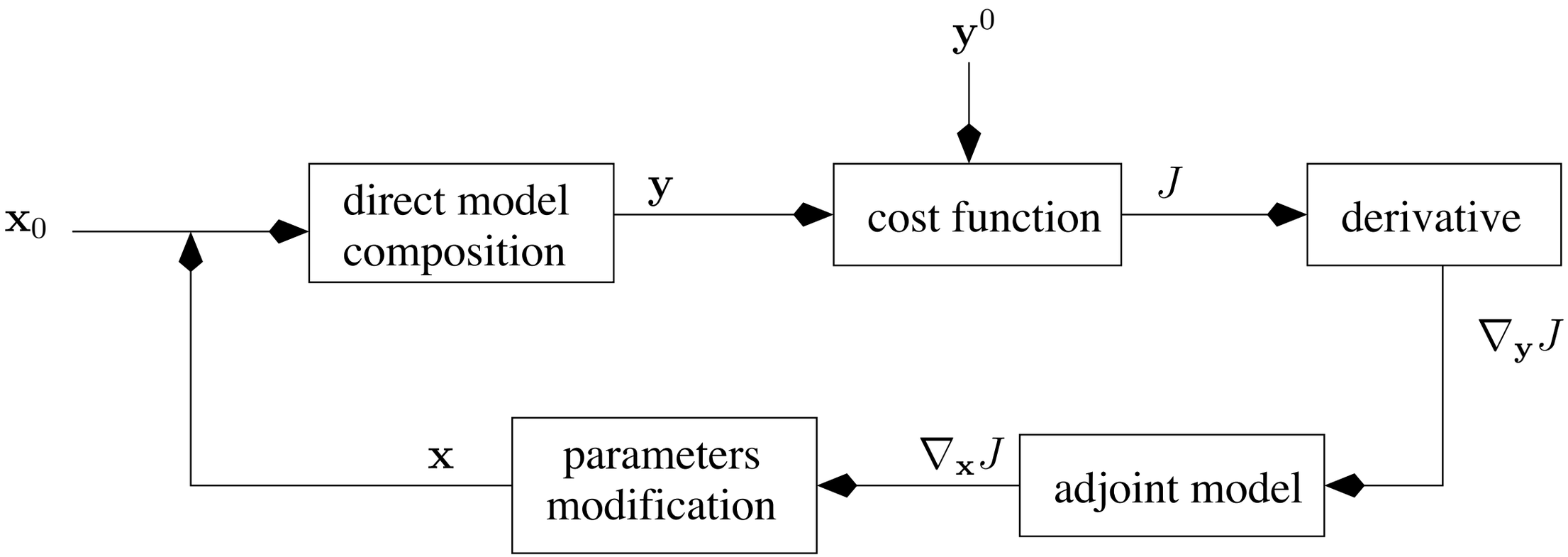
```
defval  SZX   50
defval  SZY   50
defval  SZU    1
defval  SZT  100
defval  SZA  101

hat_name  shalw

option  o_gradtest
option  o_m1qn3

traj  Toce  M  SZU  SZT

space  Soce  M  SZX  SZY  Toce

modul  Hfil   space Soce input 3 output 1 tempo cout target
modul  Ufil   space Soce input 3 output 1 tempo
modul  Vfil   space Soce input 3 output 1 tempo
modul  Hphy space Soce input 3 output 1 tempo
modul  Uphy space Soce input 3 output 1 tempo
modul  Vphy space Soce input 3 output 1 tempo

ctin  Hfil   1 from  Hfil   1  i    j    t-1
ctin  Hfil   2 from  Hphy  1  i    j    t-1
ctin  Hfil   3 from  Hphy  1  i    j    t
ctin  Ufil   1 from  Ufil   1  i    j    t-1
ctin  Ufil   2 from  Uphy  1  i    j    t-1
ctin  Ufil   3 from  Uphy  1  i    j    t
ctin  Vfil   1 from  Vfil   1  i    j    t-1
ctin  Vfil   2 from  Vphy  1  i    j    t-1
ctin  Vfil   3 from  Vphy  1  i    j    t

ctin  Hphy  1 from  Hfil   1  i    j    t-1
ctin  Hphy  2 from  Uphy  1  i -1  j    t-1
ctin  Hphy  3 from  Uphy  1  i    j    t-1
ctin  Hphy  4 from  Vphy  1  i    j    t-1
ctin  Hphy  5 from  Vphy  1  i    j+1  t-1
```

```
ctin  Uphy  1 from  Ufil   1  i    j    t-1
ctin  Uphy  2 from  Hphy  1  i    j    t-1
ctin  Uphy  3 from  Hphy  1  i+1  j    t-1
ctin  Uphy  4 from  Vphy  1  i    j    t-1
ctin  Uphy  5 from  Vphy  1  i    j+1  t-1
ctin  Uphy  6 from  Vphy  1  i+1  j    t-1
ctin  Uphy  7 from  Vphy  1  i+1  j+1  t-1

ctin  Vphy  1 from  Vfil   1  i    j    t-1
ctin  Vphy  2 from  Hphy  1  i    j-1  t-1
ctin  Vphy  3 from  Hphy  1  i    j    t-1
ctin  Vphy  4 from  Uphy  1  i-1  j-1  t-1
ctin  Vphy  5 from  Uphy  1  i-1  j    t-1
ctin  Vphy  6 from  Uphy  1  i    j-1  t-1
ctin  Vphy  7 from  Uphy  1  i    j    t-1

order modinspace  Soce
  order  YA1 YA2
         Hphy Uphy Vphy
         Hfil  Ufil  Vfil
  forder
forder
order spaceintraj  Toce
      Soce
forder

insert_fct      xdisplay
insert_fct arg  xivg
insert_fct arg  xgauss
insert_fct      xvitgeo
```

DA implementation



Variational DA pipeline



The Shallow water model

Dynamic variables :

$$u_{ijt} = \hat{u}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta x} \left[h_{i+1jt-1} - h_{ijt-1} \right] + \frac{f}{4} \left[v_{ijt-1} + v_{ij+1t-1} + v_{i+1jt-1} + v_{i+1j+1t-1} \right] - \gamma \cdot \hat{u}_{ijt-2} \right)$$

$$v_{ijt} = \hat{v}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta y} \left[h_{ijt-1} - h_{ij-1t-1} \right] - \frac{f}{4} \left[u_{i-1j-1t-1} + u_{i-1jt-1} + u_{ij-1t-1} + u_{ijt-1} \right] - \gamma \cdot \hat{v}_{ijt-2} \right)$$

$$h_{ijt} = \hat{h}_{ijt-2} - 2\Delta t \cdot H \left(\frac{u_{ijt-1} - u_{i-1jt-1}}{\Delta x} + \frac{v_{ij+1t-1} - v_{ijt-1}}{\Delta y} \right)$$

Asselin filter :

$$\begin{aligned} \hat{u}_{ijt} &= u_{ijt-1} + \alpha(\hat{u}_{ijt-1} - 2u_{ijt-1} + u_{ijt}) \\ \hat{v}_{ijt} &= v_{ijt-1} + \alpha(\hat{v}_{ijt-1} - 2v_{ijt-1} + v_{ijt}) \\ \hat{h}_{ijt} &= h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt}) \end{aligned}$$

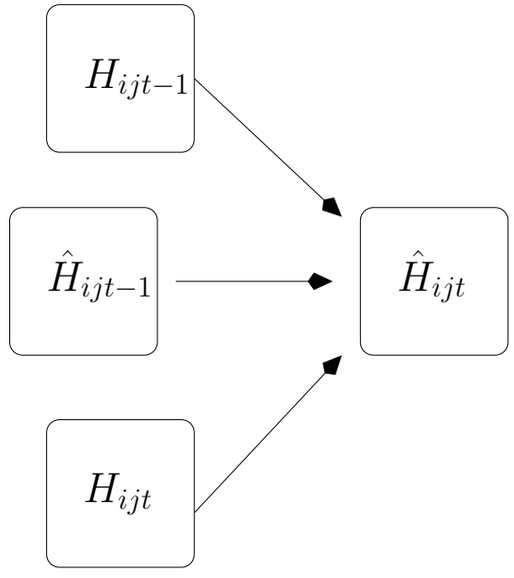
u,v : horizontal velocities

h : height of the water

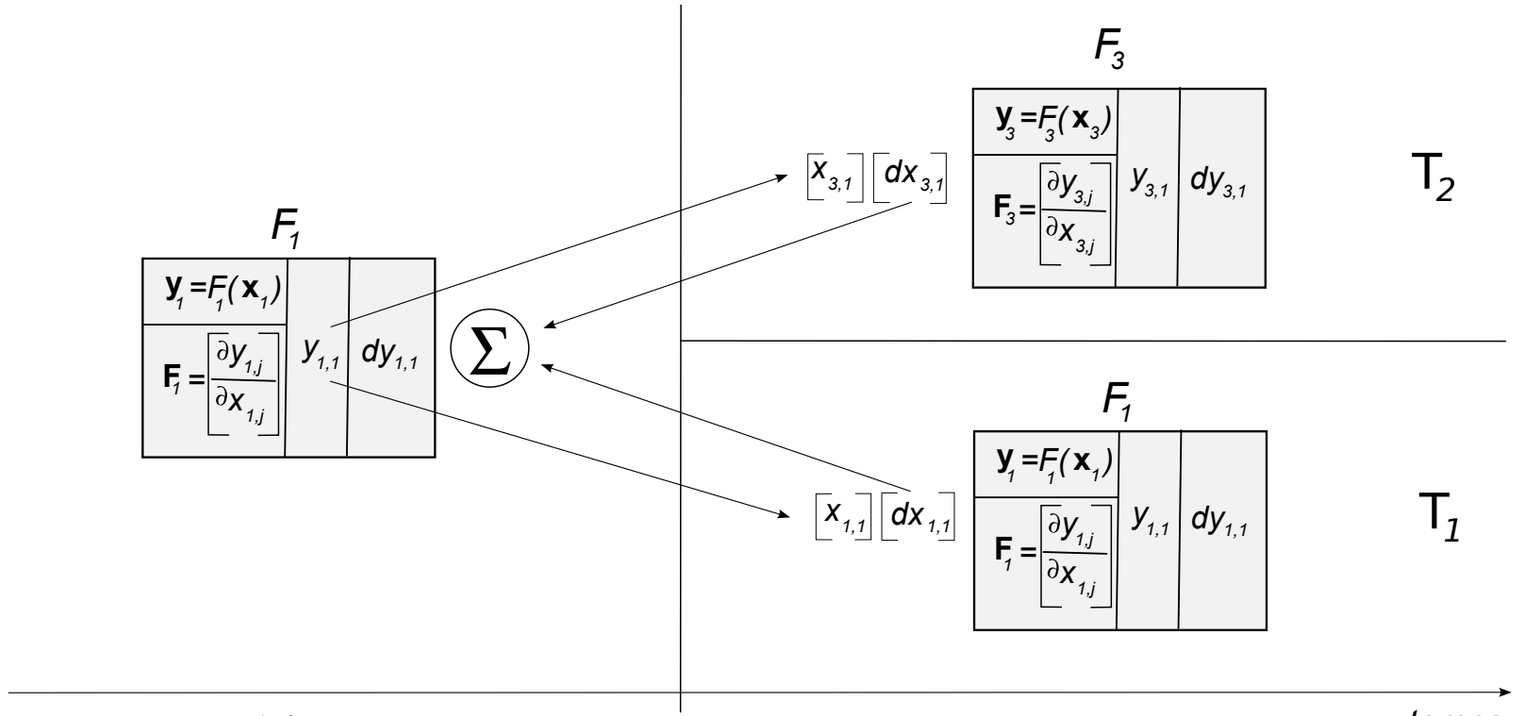
From equation to specification

```
trajectory 100
space 50 50


modul Ĥ input 3 output 1
modul Û input 3 output 1
modul V̂ input 3 output 1
modul H input 5 output 1
modul U input 7 output 1
modul V input 7 output 1

connection Ĥ from Ĥ i j t-1
connection Ĥ from H i j t-1
connection Ĥ from H i j t
connection Û from Û i j t-1
connection Û from U i j t-1
connection Û from U i j t
connection V̂ from V̂ i j t-1
connection V̂ from V i j t-1
connection V̂ from V i j t
connection H from Ĥ i j t-1
connection H from U i j t-1
...
...

order i j
    H U V Ĥ Û V̂
forder
```

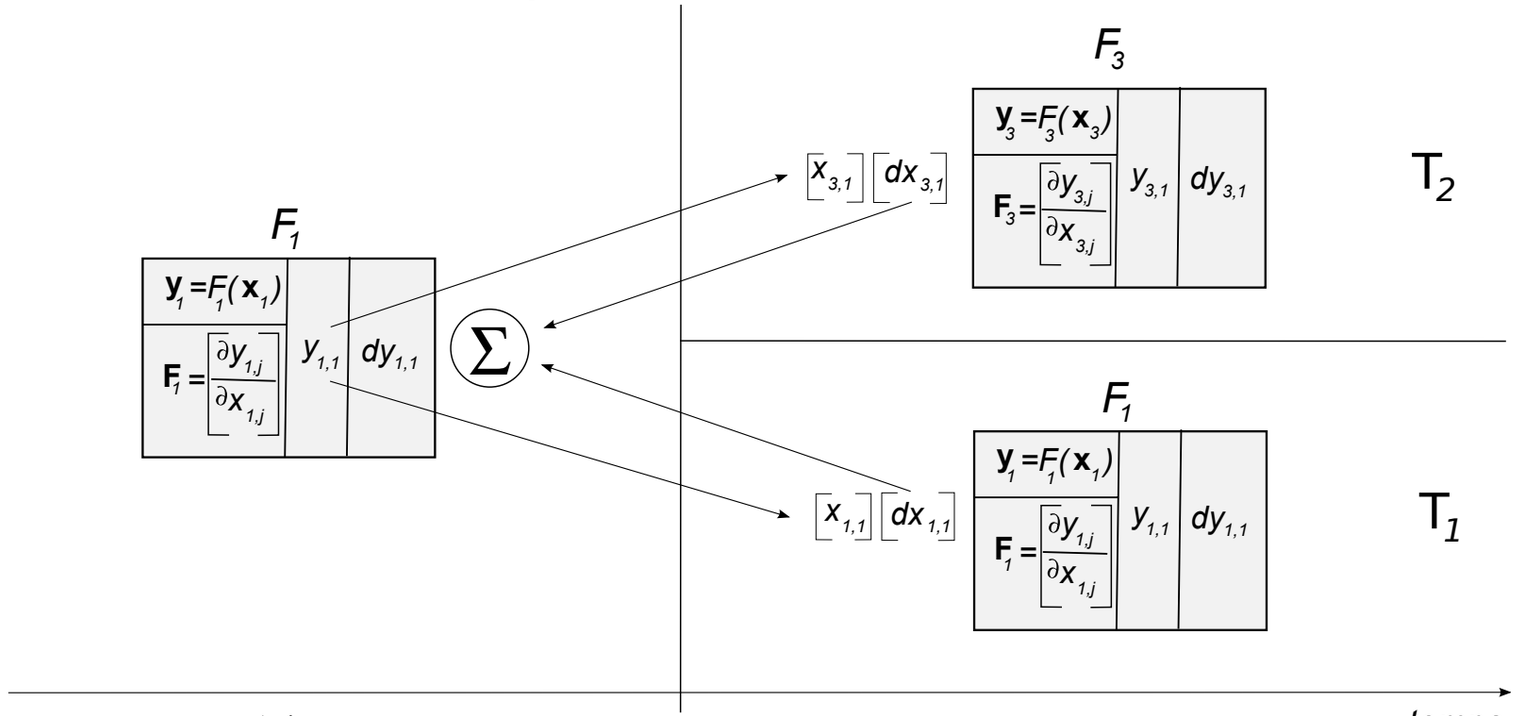
$$\hat{h}_{ijt} = h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt})$$

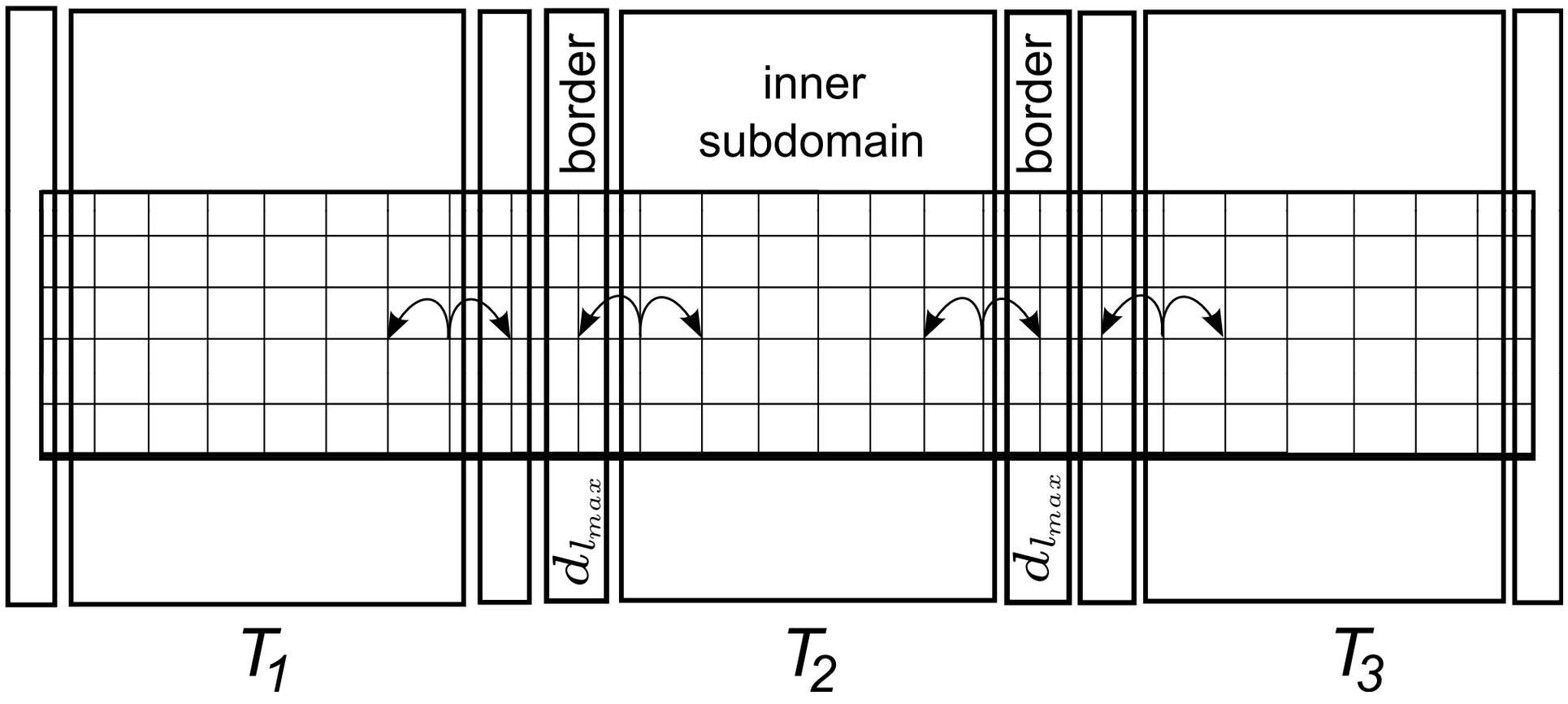


Dealing with Atomic OpenMP directives in the Backward

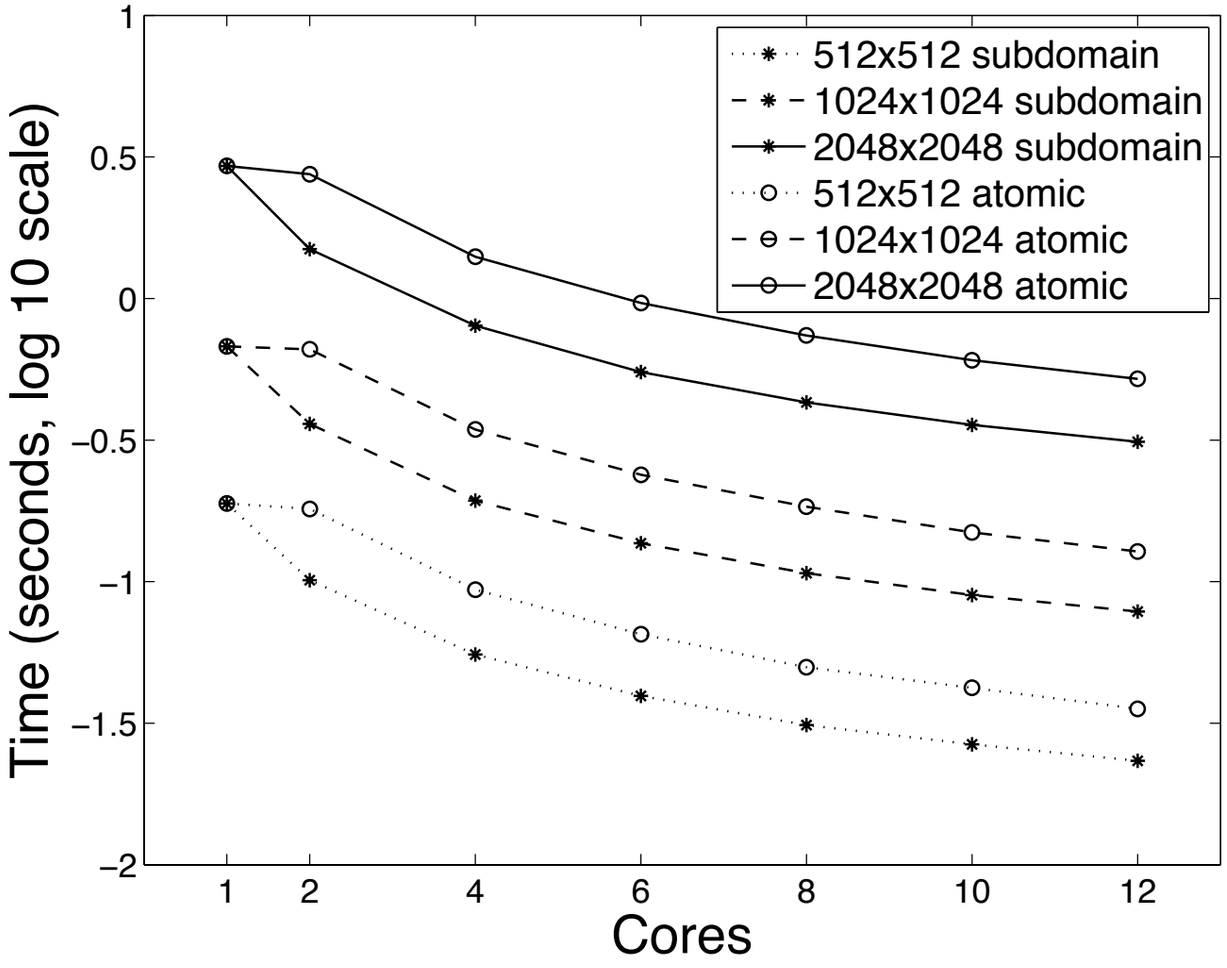


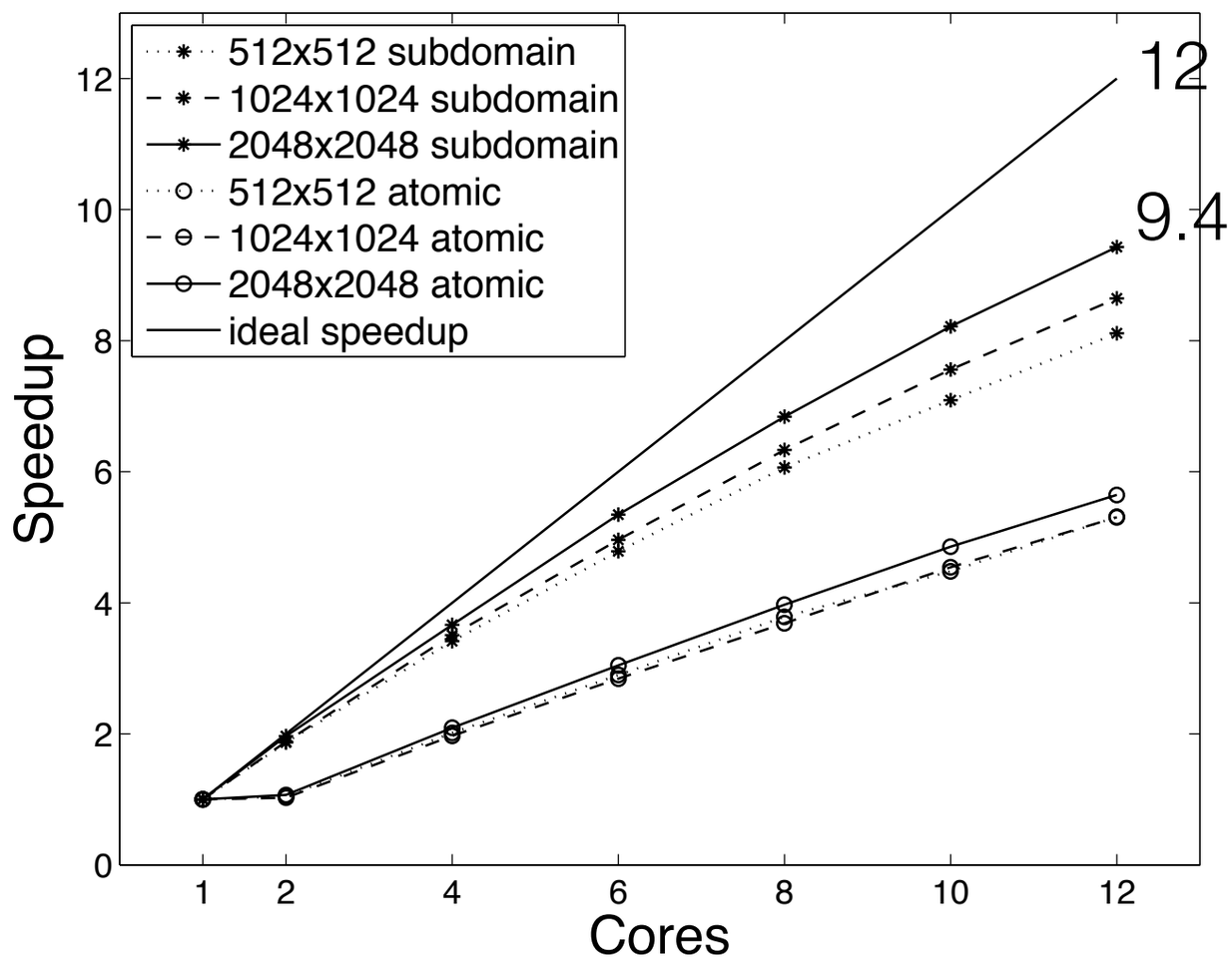
Dealing with Atomic OpenMP directives in the Backward





Performance results with OpenMP atomics: Shallow-water





Runtime over one time step

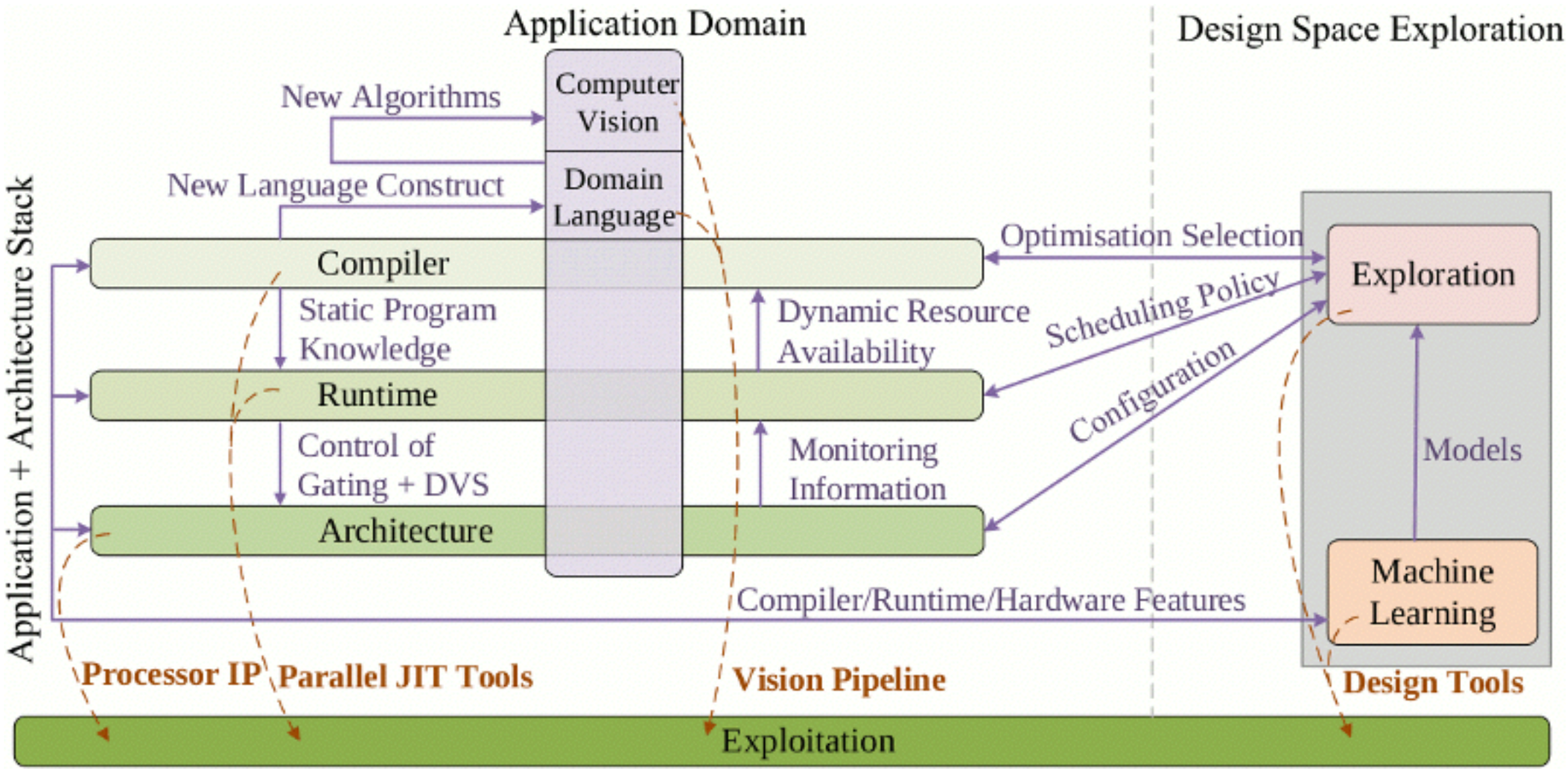
6 modules, all automatically parallelised

AMD Magny-Cours Opteron 6168:
16 GB of memory,12 cores (1.9 GHz)

PAMELA project

Panoramic Approach to the Many-corE LAndscape - from application to end-device: a holistic approach

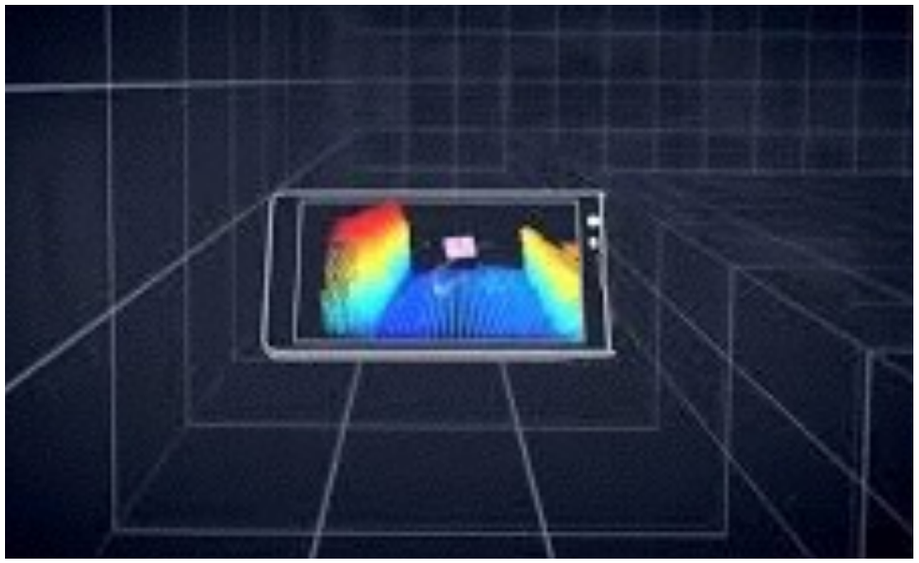
5-year EPSRC grant: Imperial, Manchester and Edinburgh



Luigi Nardi - Imperial College London

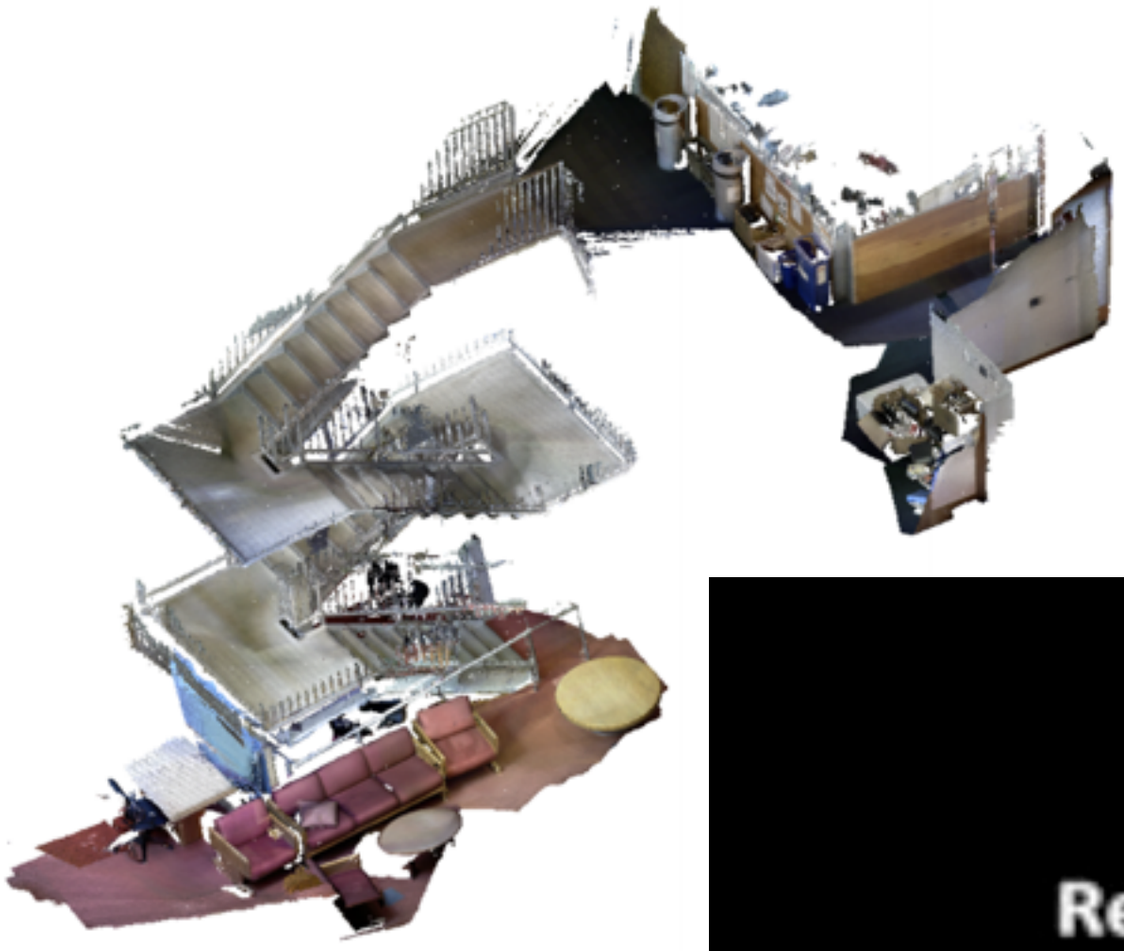
Simultaneous localisation and mapping (SLAM)

Build a coherent world representation and localise the camera in real-time



Video:
Dyson 360 Eye

Simultaneous localisation and mapping (SLAM)



[Whelan et al. 2012]

Video:
[Newcombe et al. ISMAR 2011]

Three “Performance” metrics

Holistic approach to SLAM “performance”: SLAMBench

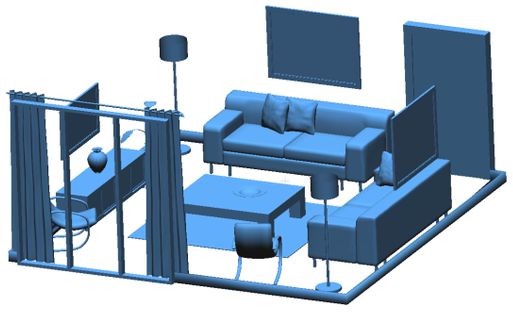
Kernels → **Architectures** → **Correctness** → **Performance** → **Metrics**

Computer Vision

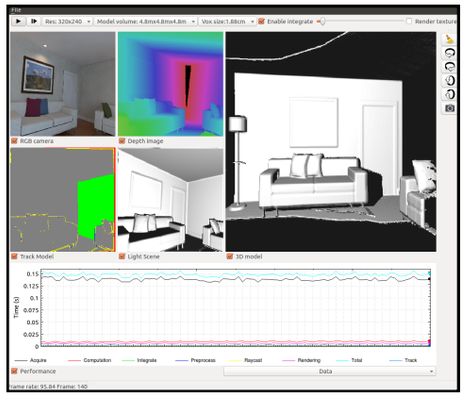
Compiler/Runtime

```
bilateralFilter (..)
halfSampleRobust (..)
renderVolume (..)
integrate (..)
      :
      :
```

Hardware



ICL-NUIM Dataset

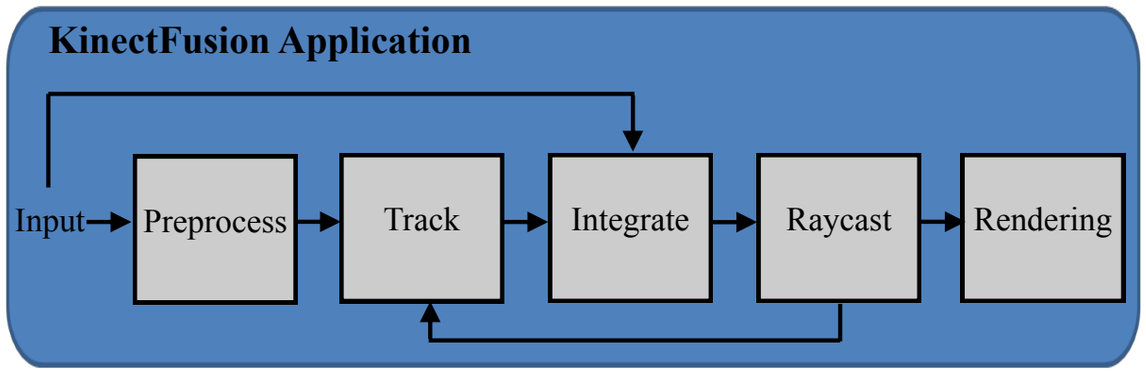


Frame rate

Energy

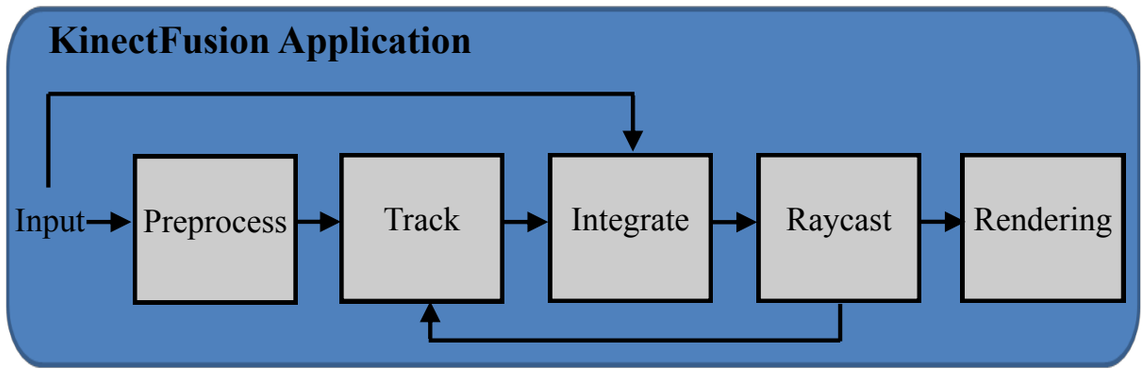
Accuracy

SLAMBench framework

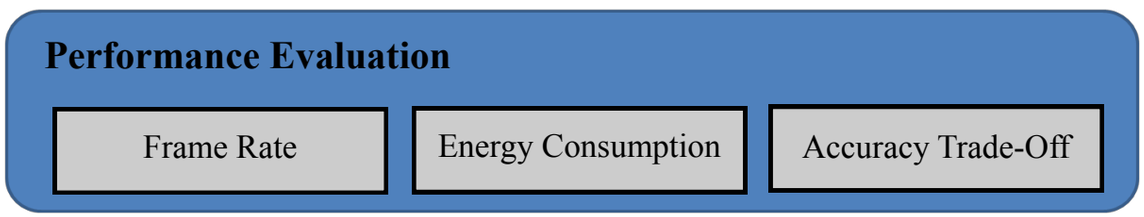


high-level blocks composed by 14 lower level kernels based on [Reitmayr 2011]

SLAMBench framework

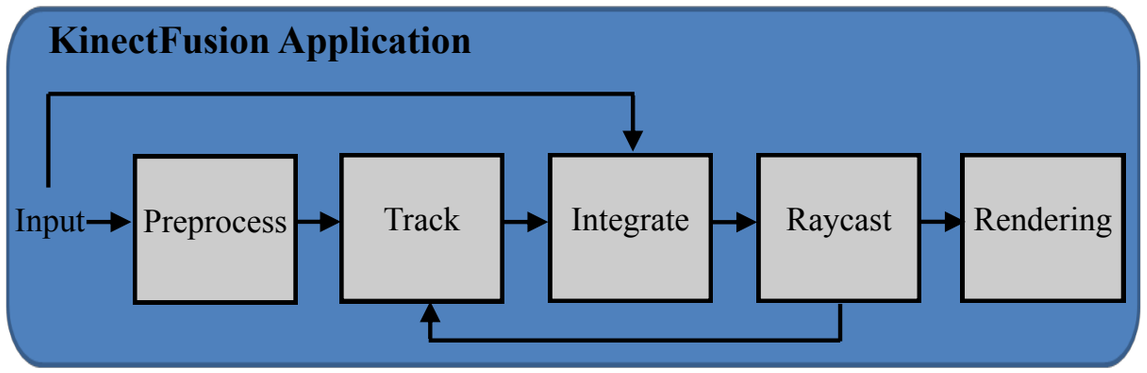


high-level blocks composed by 14 lower level kernels based on [Reitmayr 2011]

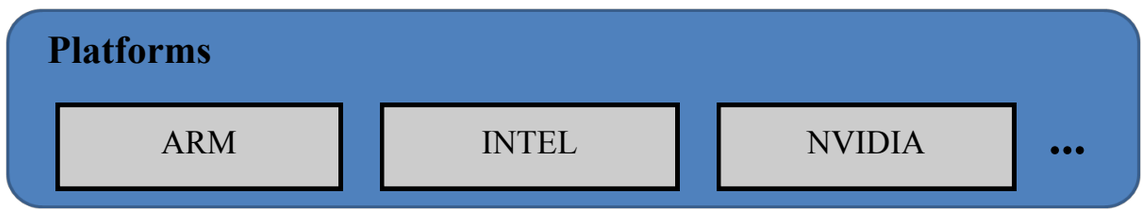


3 metrics:
execution time/energy/accuracy

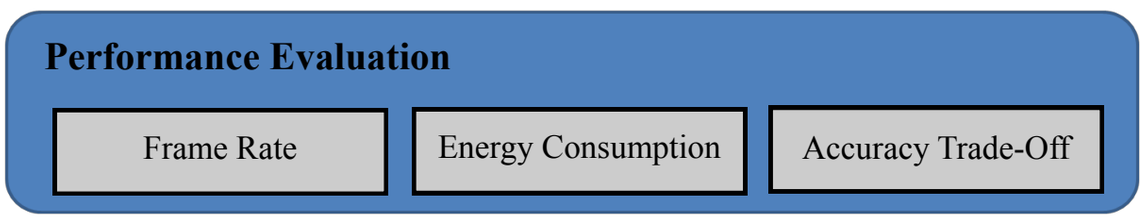
SLAMBench framework



high-level blocks composed by 14 lower level kernels based on [Reitmayr 2011]

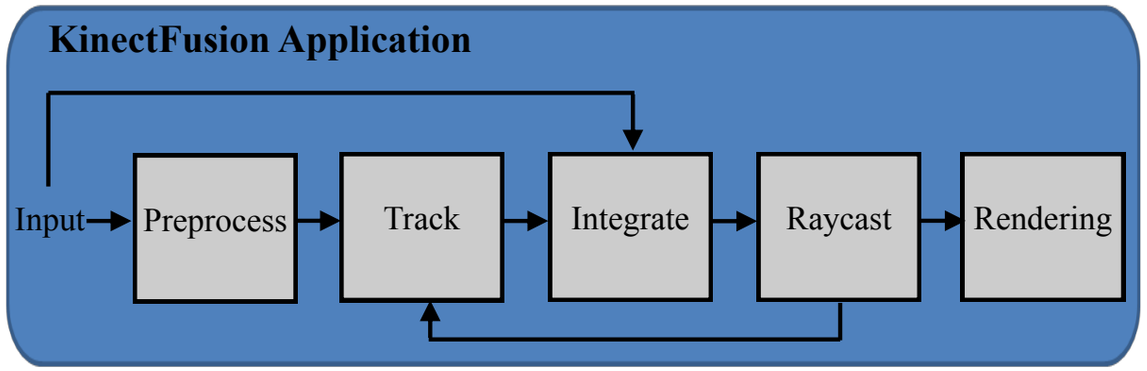


desktop, mobile, embedded: multi-core and many-core

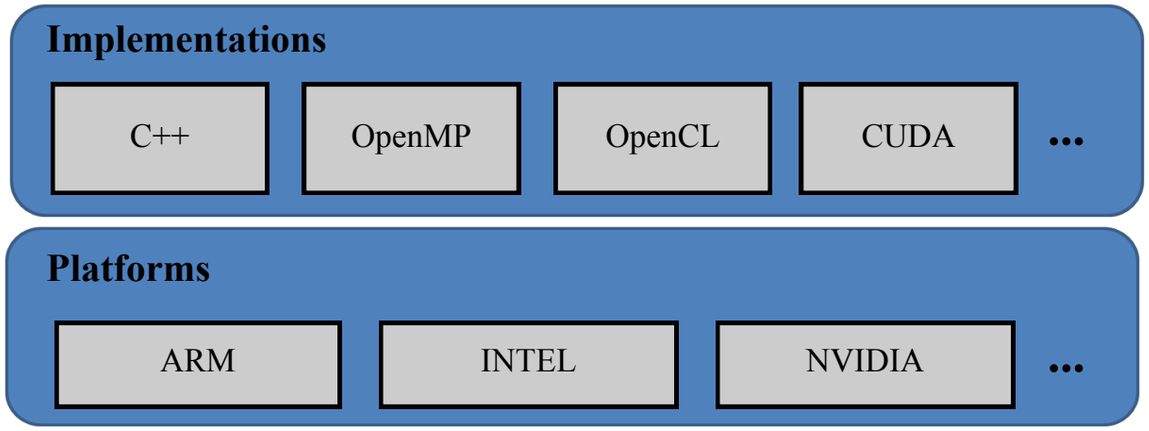


3 metrics: execution time/energy/accuracy

SLAMBench framework

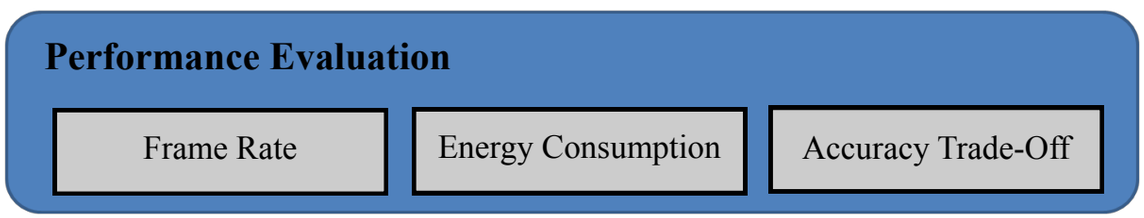


high-level blocks composed by 14 lower level kernels based on [Reitmayr 2011]



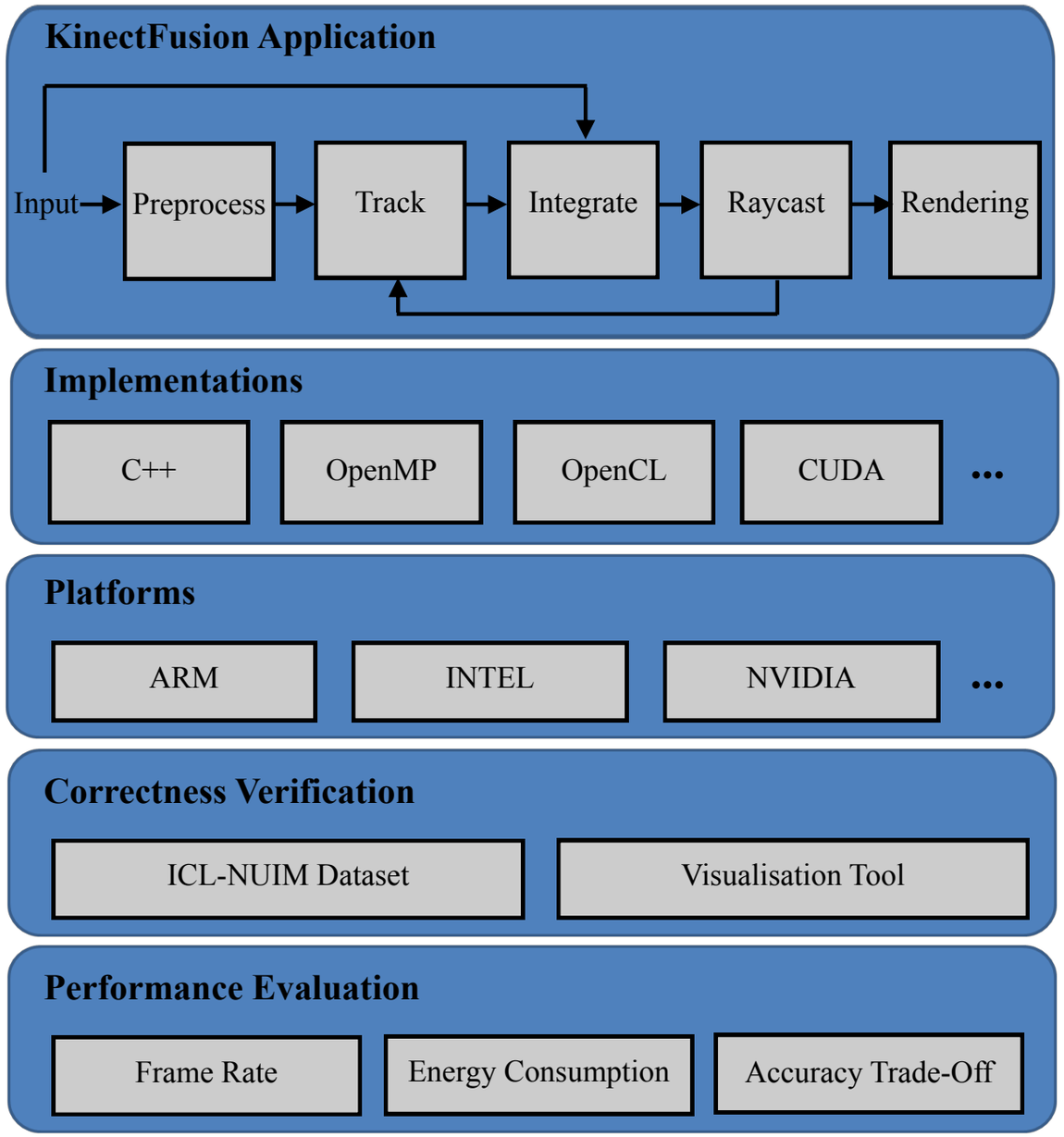
wide range of languages

desktop, mobile, embedded: multi-core and many-core



3 metrics:
execution time/energy/accuracy

SLAMBench framework



high-level blocks composed by 14 lower level kernels based on [Reitmayr 2011]

wide range of languages

desktop, mobile, embedded: multi-core and many-core

integrated verification/visualisation

3 metrics: execution time/energy/accuracy

Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

Platforms

Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

Platforms

Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

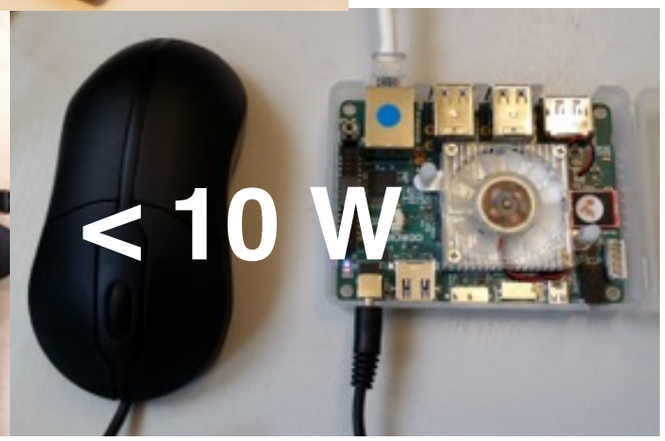
Platforms

Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

Platforms

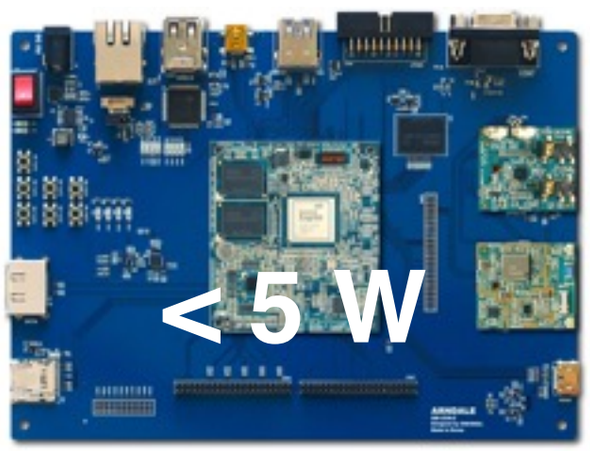
Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

Platforms



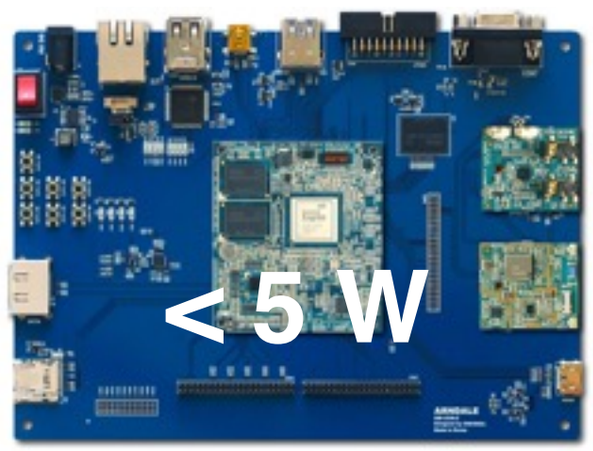
Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

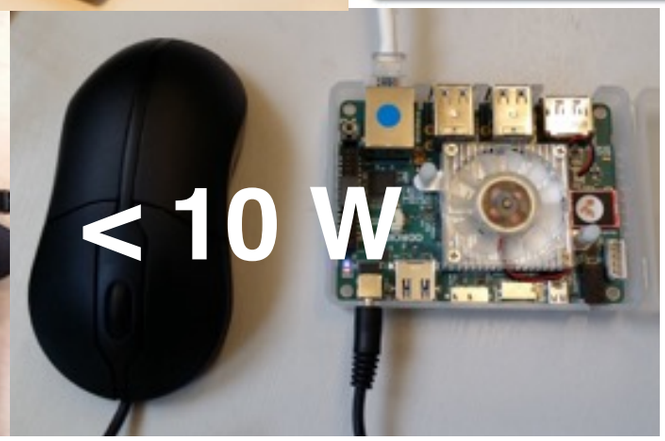
Platforms



Machines	TITAN	GTX870M	TK1	ODROID	Arndale
CPU	Intel	Intel	ARM	ARM	ARM
CPU name	i7 Haswell	i7 Haswell	NVIDIA 4-Plus-1	Exynos 5422	Exynos 5250
CPU GFLOPS	448	307	74	80	27
CPU cores	4	4	4 + 1	4 + 4	2
GPU	NVIDIA	NVIDIA	NVIDIA	ARM	ARM
GPU name	TITAN	GTX 870M	Tegra K1	Mali-T628	Mali-T604
GPU GFLOPS	4500	2520	330	60 + 30	60

Platforms





“Performance”: accuracy

Absolute trajectory error (ATE) in cm - default algorithmic configuration

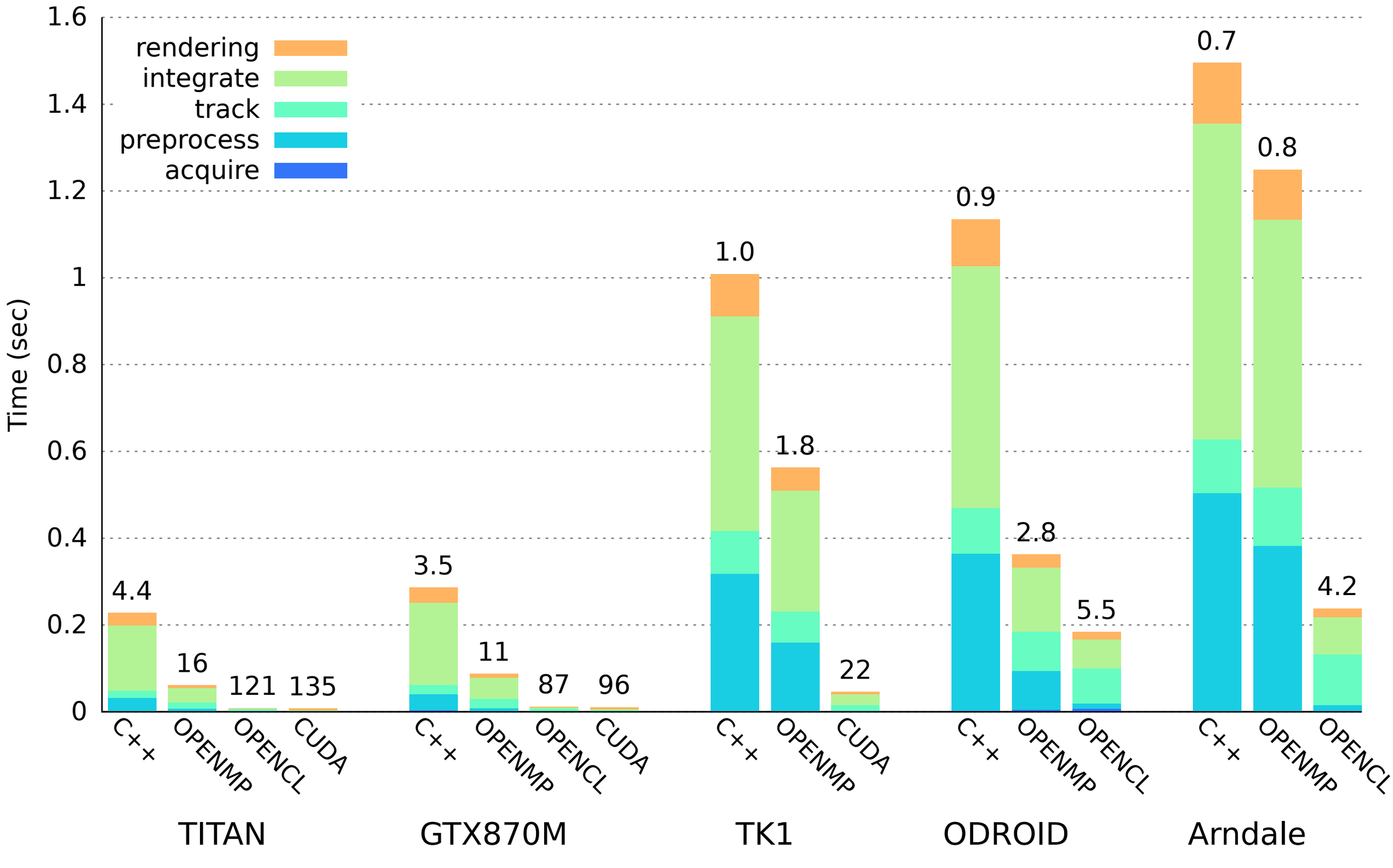
ATE in cm	TITAN	GTX870M	TK1	ODROID	Arndale
C++	2.07	2.07	2.06	2.06	2.06
OpenMP	2.07	2.07	2.06	2.06	2.06
OpenCL	2.07	2.07	n/a	2.01	2.07
CUDA	2.07	2.07	2.07	n/a	n/a

- ATE easy-to-use tool for non computer vision experts
- Semantic validation instead than bitwise accuracy

Luigi Nardi - Imperial College London

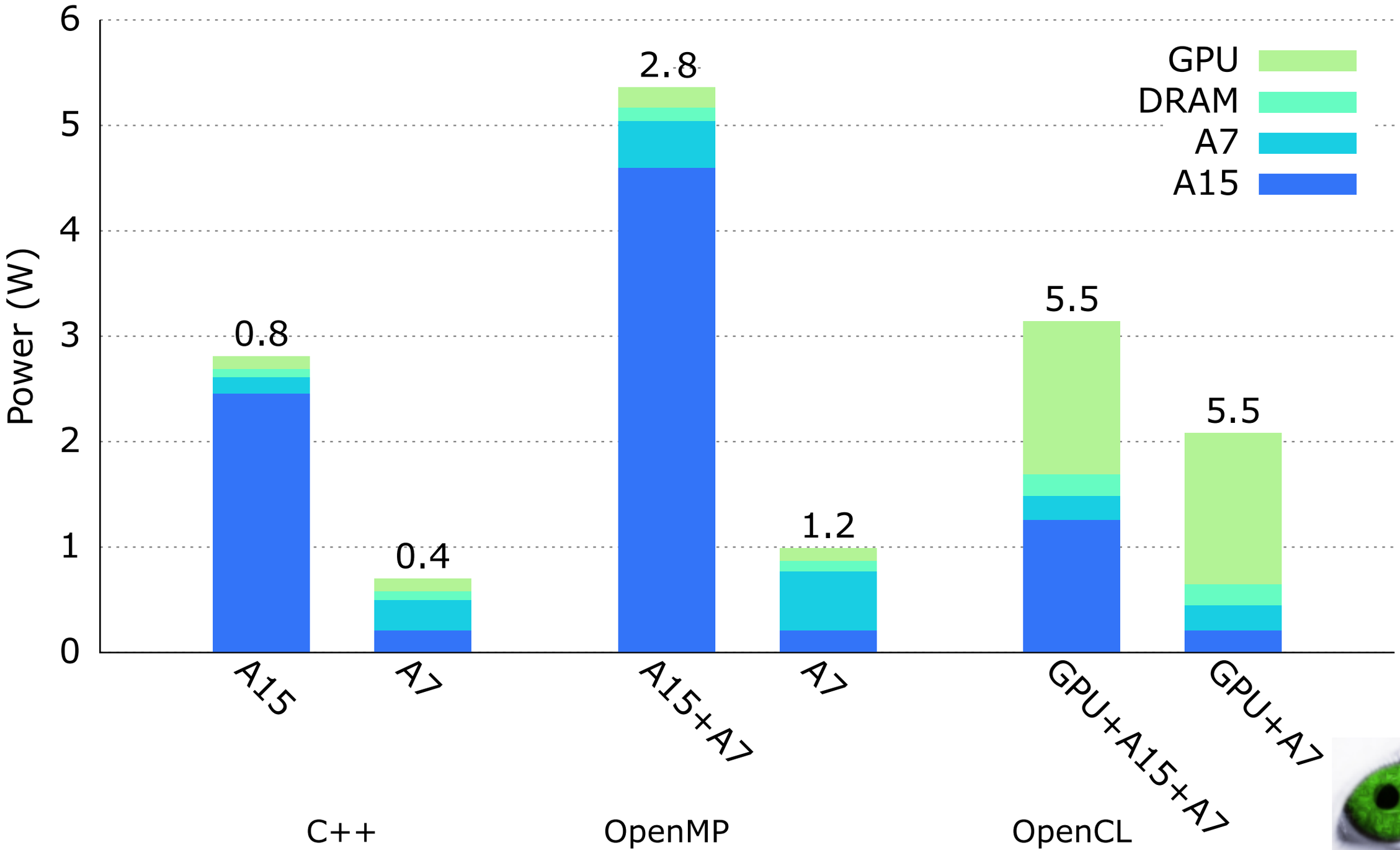
“Performance”: execution time

Mean time per frame (lower is better)



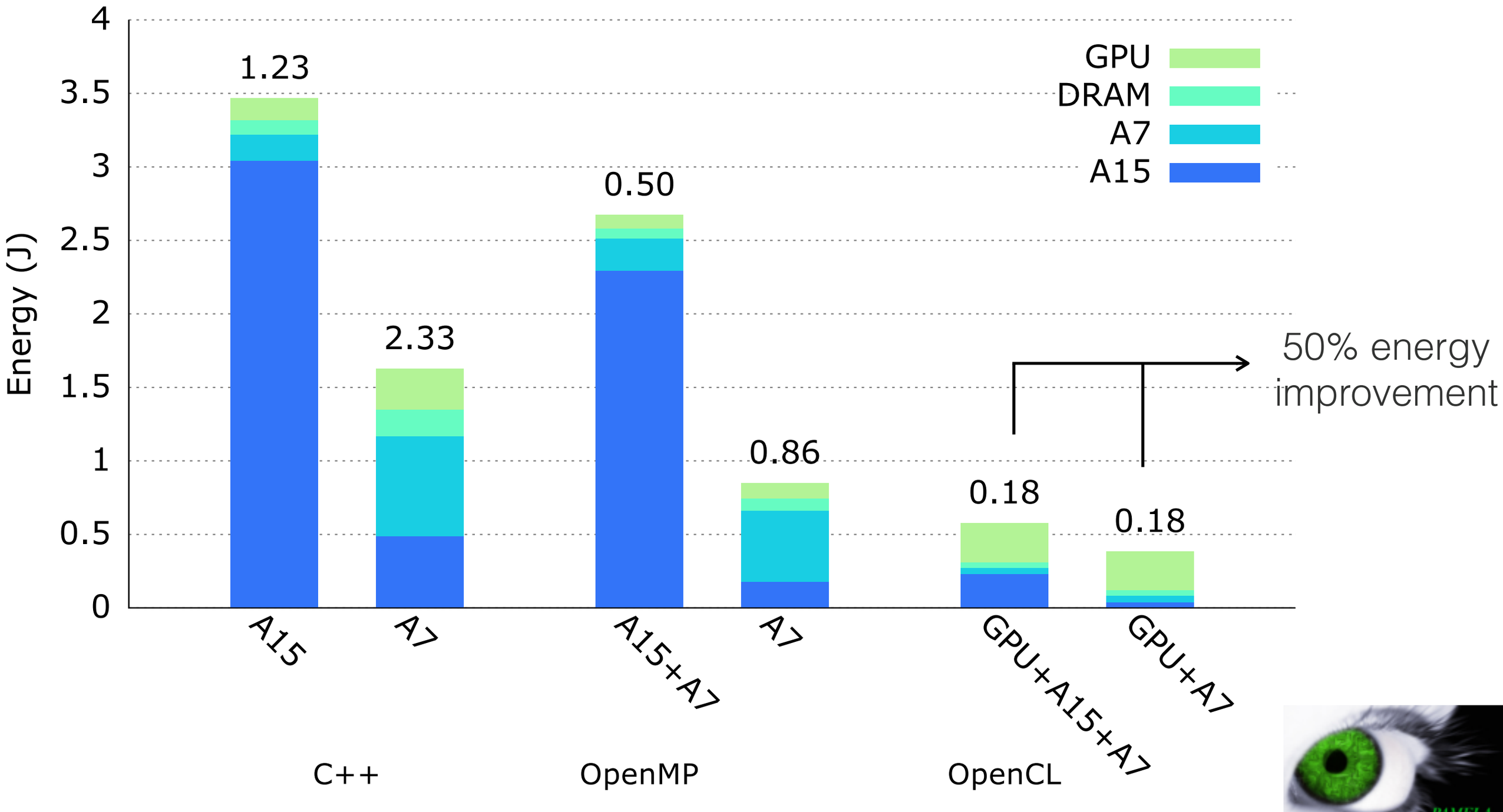
“Performance” power (ODROID-XU3)

On-board voltage/current sensors and split power rails:
power measured individually on big (A15), LITTLE (A7), GPU and DRAM



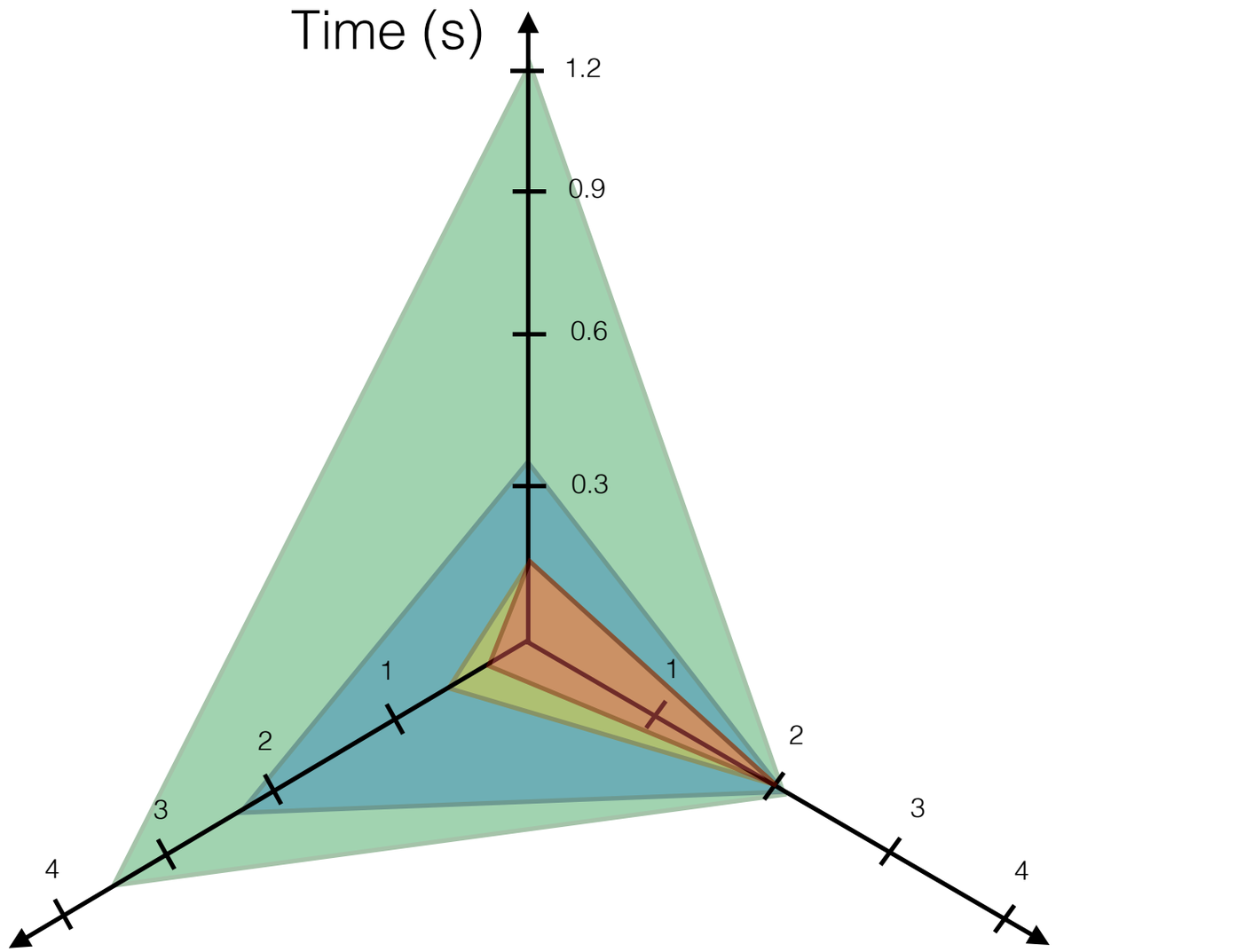
“Performance”: energy (ODROID-XU3)

Energy usage per frame, mean time as marked



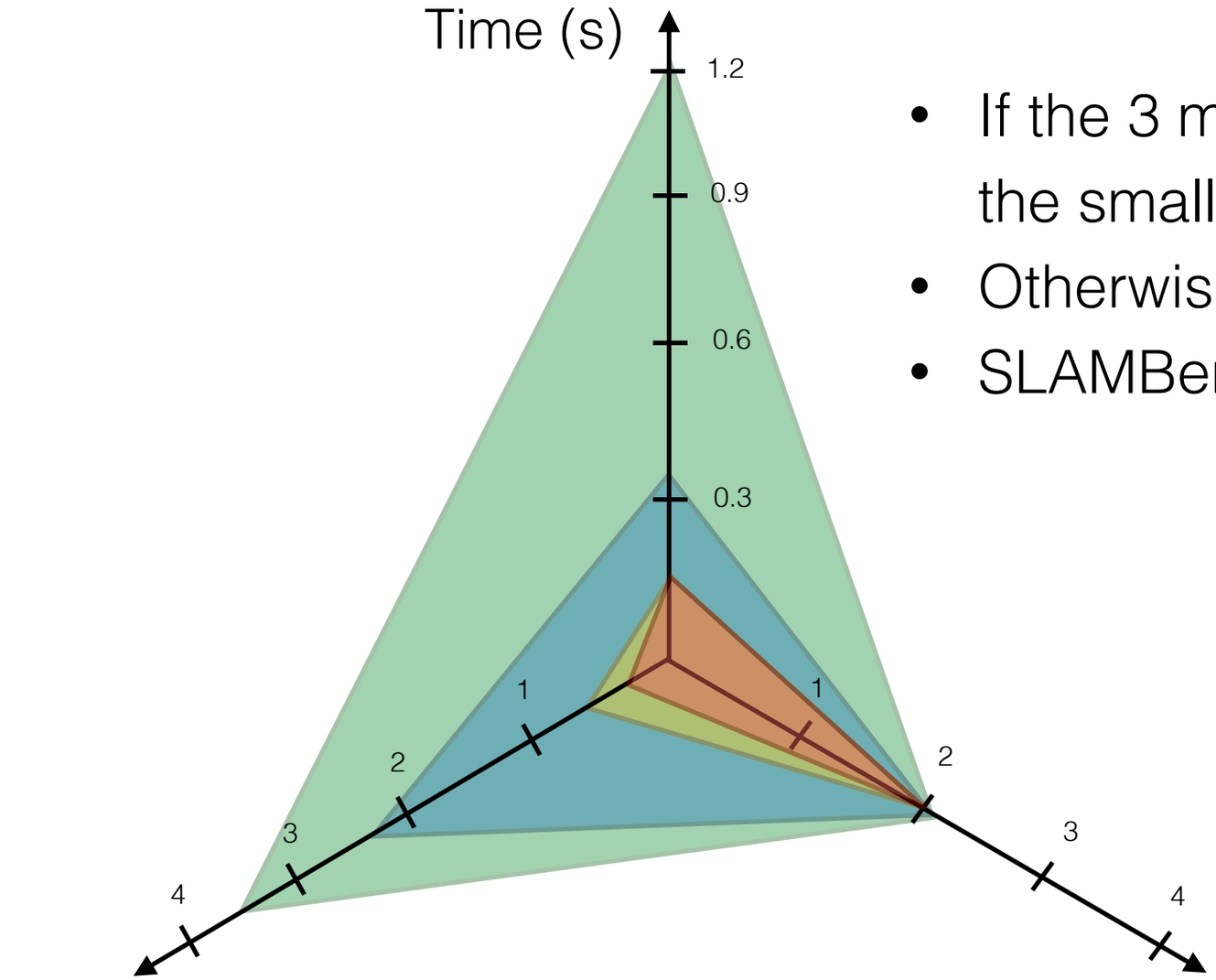
“Performance” trade-off

Language	Configuration	Error (cm)	Time (s)	Time (FPS)	Energy (J)	Area	Colour
C++	A15	2.06	1.23	0.81	3.47	4.4	
OpenMP	A15 + A7	2.06	0.35	2.8	2.67	2.8	
OpenCL	GPU + A15 + A7	**2.01**	**0.18**	5.5	**0.58**	0.6	
OpenCL	GPU + A7	**2.01**	**0.18**	5.5	**0.38**	0.4	



“Performance” trade-off

Language	Configuration	Error (cm)	Time (s)	Time (FPS)	Energy (J)	Area	Colour
C++	A15	2.06	1.23	0.81	3.47	4.4	
OpenMP	A15 + A7	2.06	0.35	2.8	2.67	2.8	
OpenCL	GPU + A15 + A7	**2.01**	**0.18**	5.5	**0.58**	0.6	
OpenCL	GPU + A7	**2.01**	**0.18**	5.5	**0.38**	0.4	



- If the 3 metrics are equally important: the smaller area the better: 4.4 > 2.8 > 0.6 > 0.4
- Otherwise weighted area
- SLAMBench use case: 52% energy improvement

SLAMBench today: company users

SLAMBench today: academic users

Copyrights

- Author: unknown. Microsoft Kinect camera. [Image]. Retrieved from http://channel9.msdn.com/Series/KinectSDKQuickstarts/Understanding-Kinect-Hardware
- Author: Dyson Ltd. Dyson 360 Eye. [Photograph]. Retrieved from http://www.blessthisstuff.com/stuff/technology/misc-gadgets/dyson-360-eye/
- Author: Google Inc. Google Tango project. [Image]. Retrieved from http://blogthinkbig.com/en/project-tango-googles-mobile-kinect/
- Author: unknown. Audi autonomous car. [Photograph]. Retrieved from http://www.wired.com/2010/06/audis-robotic-car-looks-hot-in-old-school-livery/
- Author: ExtremeTech. Google Shaft robot. [Photograph]. Retrieved from http://www.extremetech.com/extreme/173318-google-wins-darpas-robotics-challenge-wonders-if-it-was-a-good-idea-to-turn-down-future-military-contracts
- Author: HardKernel. ODROID-XU3 board. [Photograph]. Retrieved from http://www.hardkernel.com/main/products/prdt_info.php?g_code=G135235611947
- Author: PC Specialist Ltd. Vortex series laptop. [Photograph]. Retrieved from https://www.pcspecialist.co.uk/forums/showthread.php?23366-My-new-beast-15-6-quot-Vortex-III
- Author: Arndale.org. Arndale board. [Photograph]. Retrieved from http://www.arndaleboard.org/wiki/index.php/Main_Page
- Author: lizwang. Black-Scholes option pricing model. [Image]. Retrieved from https://lizwang.wordpress.com/2009/09/01/black-scholes-option-pricing-model/
- Author: Unknown. NVIDIA chip. [Image]. Retrieved from http://www.nvidia.com/object/tesla-workstations.html
- Author: Unknown. INTEL chip. [Image]. Retrieved from http://www.newegg.com/Product/Product.aspx?Item=N82E16819117234
- Author: Unknown. SPARC chip. [Image]. Retrieved from http://en.wikipedia.org/wiki/SPARC_T5
- Author: snugsomeone. Labyrinth. [Image]. Retrieved from http://snugsomeone.deviantart.com/art/Labyrinth-111756988