Elastic Algorithmic Skeletons into Distributed Systems

Sonia Campa[1], Marco Danelutto[1], Horacio González-Vélez[2],
Alina Madalina Popescu[2] and Massimo Torquati[1]

(1) Computer Science Department, University of Pisa Italy
E-mail: {campa, marcod, torquati}@di.unipi.it
(2) Cloud Competency Center, National College of Ireland, Ireland
E-mail: {Alina-madalina.Popescu, horacio}@ncirl.ie

DUBLIN
10 June 2014

Background

- Latency
  - Hierarchical Memory – How many cycles do I need to?
  - File Sizes? SneakerNet?
- Resources are finite
  - 32 bit vs 64 bit? Max Matrix Size?
  - Local Cores ?
  - Specialised Units ?
  - MakeSpan? Power? Other?

RESOURCES or LATENCY ?

Use the Cloud

Textbook Definition

Computational Resources* where the Boundaries are determined by **Economic Factors** rather than Technical Limits

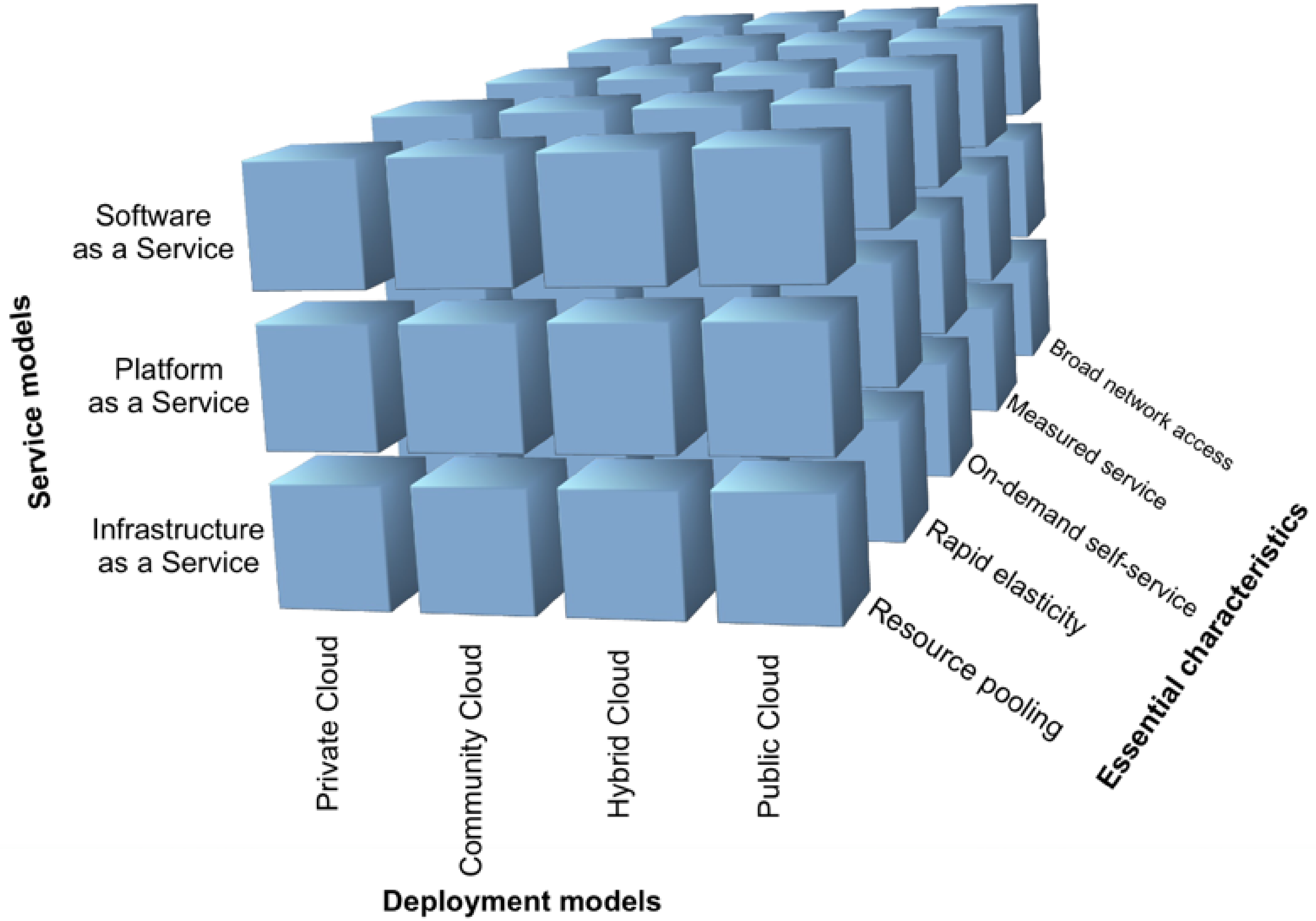
On-demand Computing

Software as a Service

Internet as a Platform

*N.B. Computational resource is commonly used to describe accessible computing equipment and software.

Cloud 101



 Credits: Kate Craig-Wood, NIST, and G-Cloud

Is it HPC?
Parallel ?
DSM? DM? D?

Concepts

- Main goal
  - Deploy FastFlow on virtual multi-core machine and cluster of virtual multi-core machines.

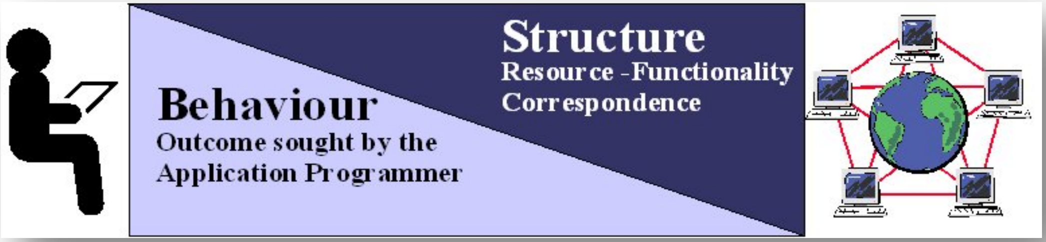
- Our contribution
  a. Virtualisation overhead measurement in the KVM environment
  b. FastFlow deployment and testing on the Amazon EC2 public cloud infrastructure

Algorithmic Skeletons

Higher-Order Functions

Abstract and Implement **Patterns** of Parallel Computation, Communication, and Interaction

Decouple Behaviour (Computation) from Structure (Coordination)



Algorithmic Skeletons

Skeleton	Scope	Example
Data-Parallel	Data Structures	Scan, Map, Broadcast, Reduce, Gather, Scatter,
Task-Parallel	Tasks	Farm, Pipeline, …
Resolution	Family of Problems	Div &Conq, Br & Bnd, Dyn Prog, Heuristic Opt, …

http

Pattern or Skeleton?

- Skeleton: Defines a parallel pattern in terms of computational nodes, data and control dependencies

Parallel Pattern

=

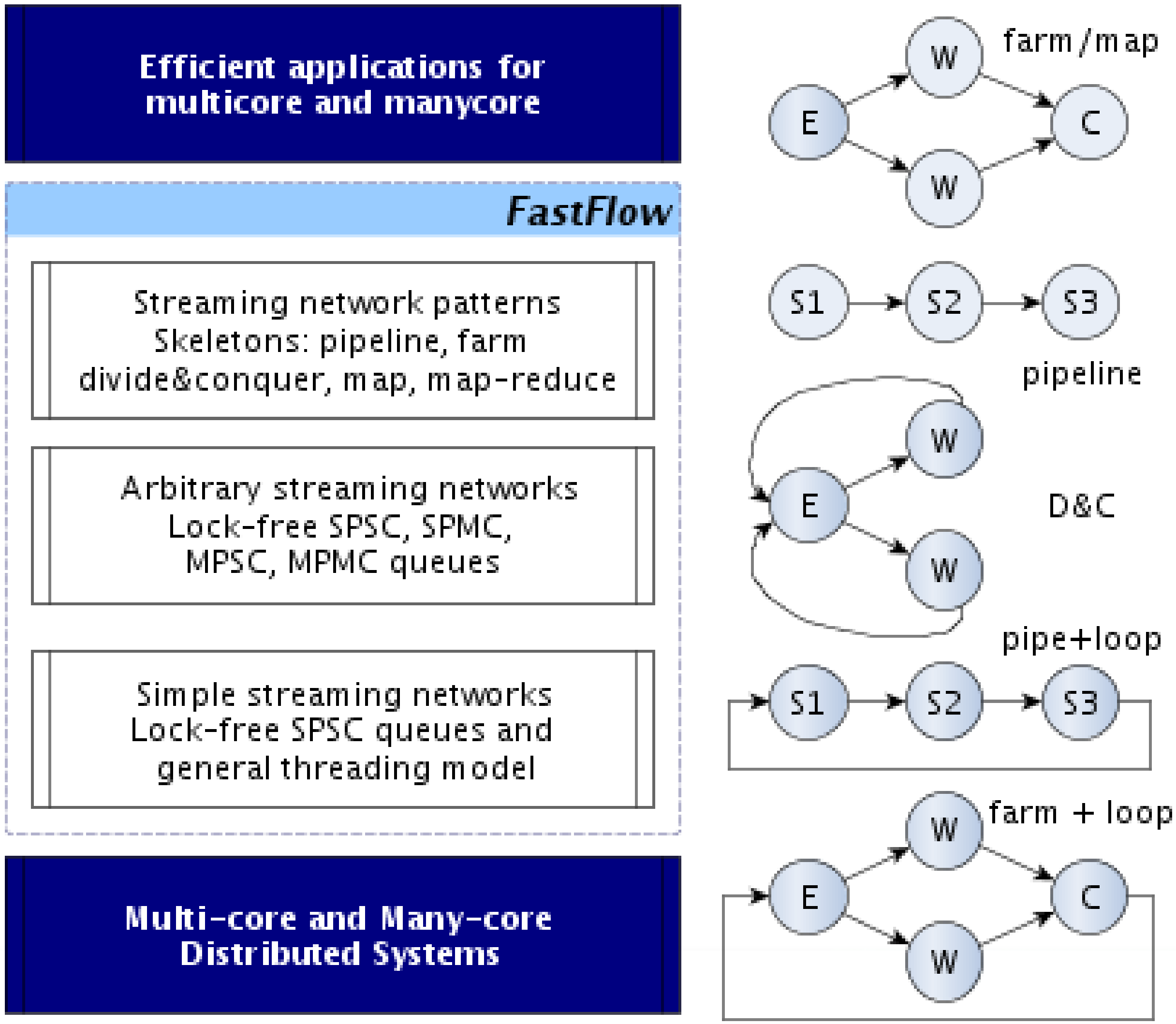
Algorithmic Skeleton + GoF SE Req's

- Aim: **Write the application using skeletons once and deploy "everywhere"**
  - Application and **Performance Portability**
    - Run-time support to cope with low-level platform details

FastFlow concepts

- Structured parallel programming framework
- FastFlow: Skeletons = C++ classes & templates (via Pthreads).
- Target: Multi-core CPU, Dist Sys, GPU
- Stream parallel patterns: pipeline, task-farm, loopback
  - Ongoing work for map and map-reduce skeletons on multi-core
- Task-offloading on Tile64 and GPUs
- ParaPhrase Programming Framework

FastFlow concepts



Virtual Platforms

- Virtualisation 101
  - Multiple OS on a single physical machine
  - Partitioning of resources (power efficiency)
  - Security
    - Reset the VM after a job has finished
  - Checkpoint & Restart
  - Dynamic Configuration Changes
    - Load a different VM vs reinstalling
    - Add/delete virtual cores, disk space, NICs,….

Virtual Platform for HPC

- HPC support via System-level virtualisation techniques (hw)
- Overhead
  - Overhead cannot be statically quantified in advance
- Overhead is compensated via
  - more virtual resources and favourable **performance / cost** ratio
  - application dependant

Eucalyptus private cloud

- Implementation Environment
  - **Single VM Є Eucalyptus** Private cloud @ NCI
  - Linux **KVM** virtual environment
  - 2 6-core CPUs Intel Xeon E6-2540 @2.5 GHz, with 8MB L3 with Linux CentOS x86_64
- VM (Linux CentOS x86_64) has 6 cores
  - Forced to be executed on a single socket of the physical machine

Eucalyptus private cloud

Sequential benchmark

- Micro-benchmark: the Square Matrix Multiplication (MatMul)
- 2 versions: Standard & Cache-oblivious

Cache-Oblivious Algorithm:

```
double A[N][N],B[N][N],C[N][N];
 for(i=0;i<N;++i)
    for(j=0;j<N;++j)
       for(k=0;k<N;++k)
          C[j][k] += A[j][i]*B[i][k];
```

Standard Algorithm:

```
double A[N][N],B[N][N],C[N][N]
 for(i=0;i<N;++i)
    for(j=0;j<N;++j)
       for(k=0;k<N;++k)
          C[i][j] += A[i][k]*B[k][j];
```

Overhead: sequential run

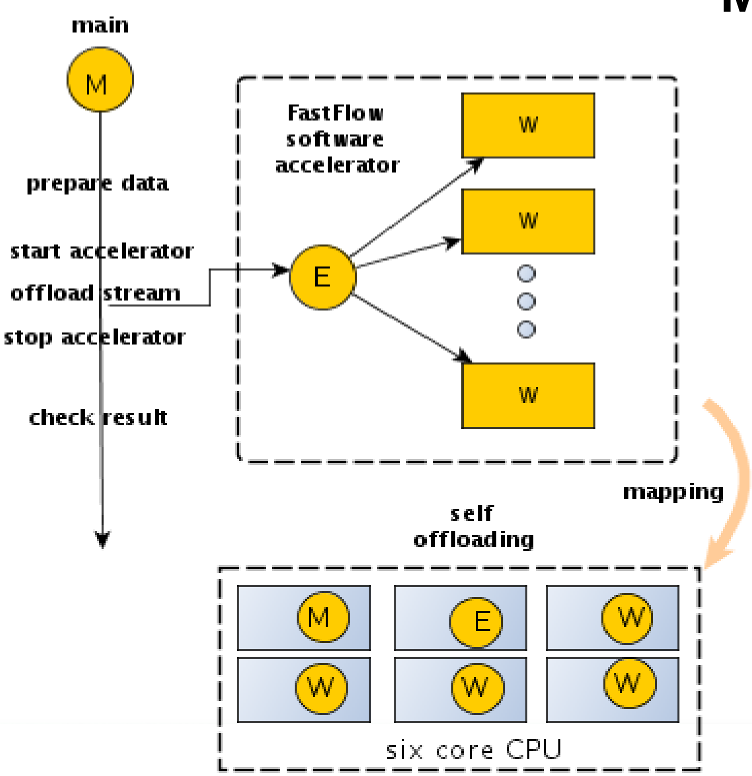
Time in Sec. (approximated)	512		1024		2048	
	P	V	P	V	P	V
Standard	0.34	0.36	2.4	2.59	110.8	119.2
Cache-Oblivious	0.15	0.19	0.68	0.76	8.5	9.4

P= Physical V=Virtual

MatMul Overhead	512	1024	2048
Standard	5.96%	6.01%	7.48%
Cache-Oblivious	27.43%	11.52%	10.29%

Parallel benchmark

- As parallel benchmark we use again Matrix Multiplication Implemented using FastFlow software accelerator:



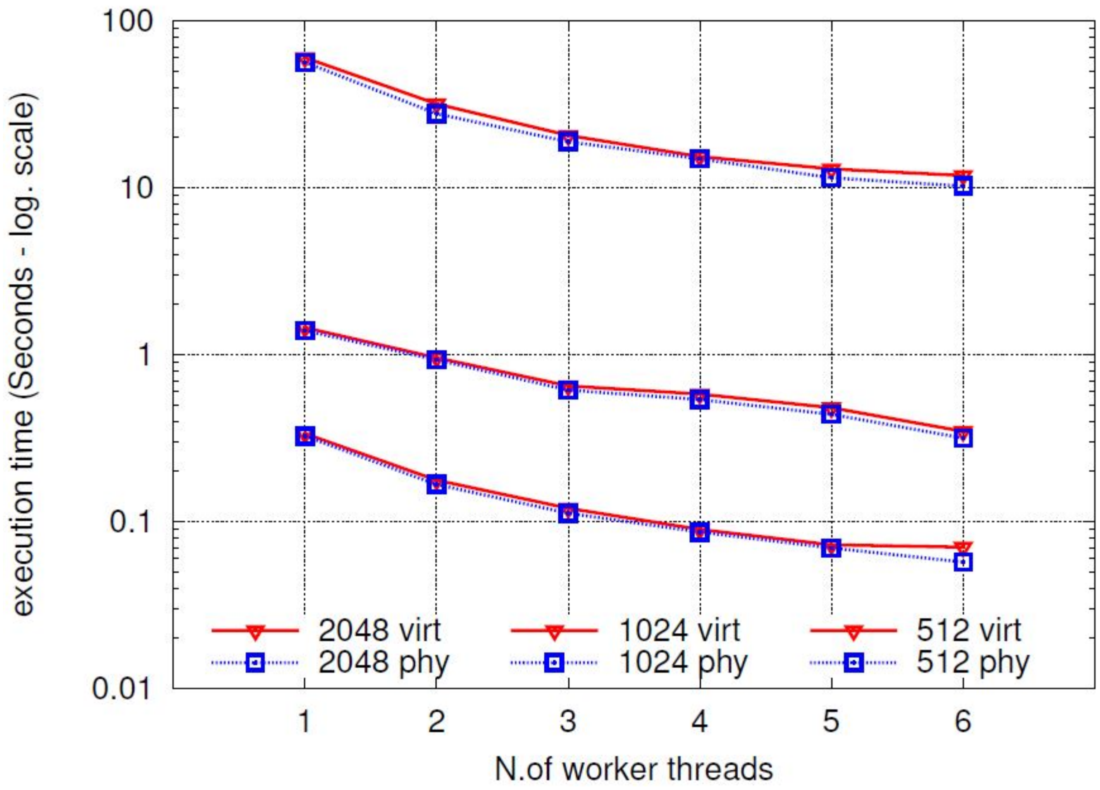
Main code:

```
double A[N][N],B[N][N],C[N][N]
 for(index=0;index<N;++index)
    farm.offload(index);
```

Worker code:

```
<for each input idx value>
 for(j=0;j<N;++j)
    for(k=0;k<N++k)
       C[idx][j] += A[idx][k]*B[k][j];
```

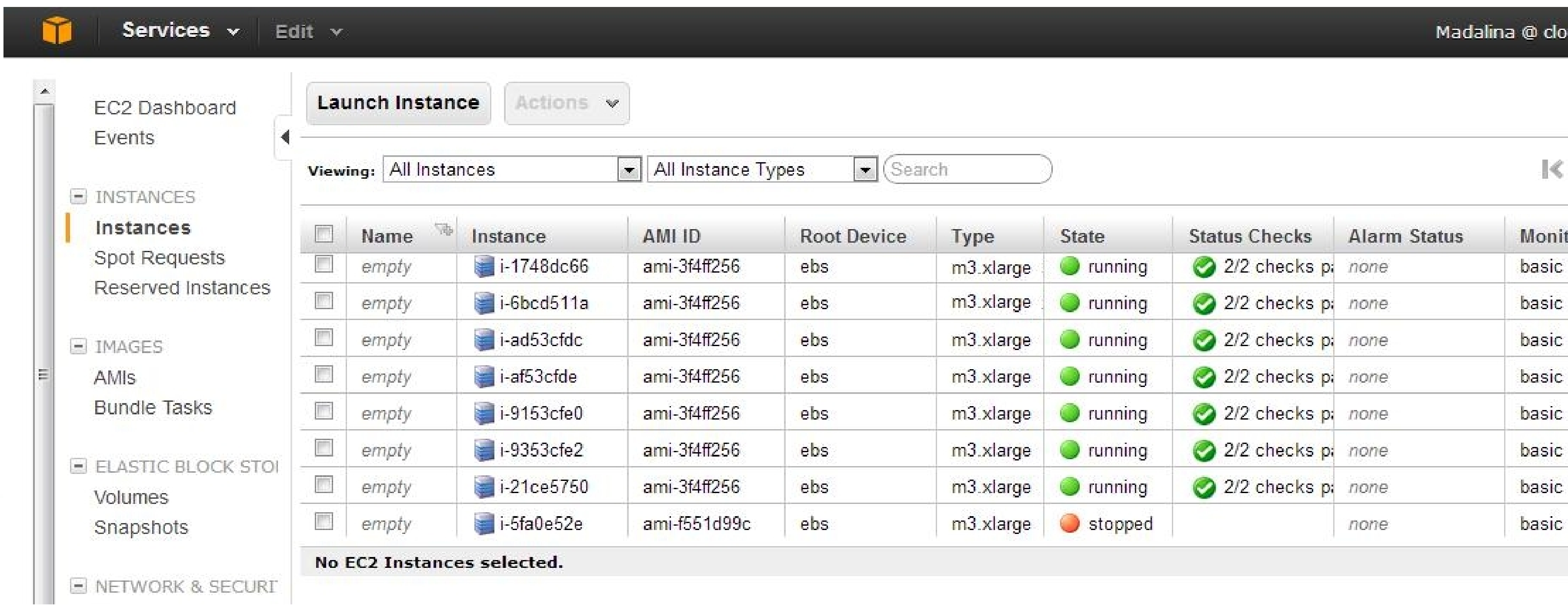
Overhead: parallel run



Parallel MatMul	512	1024	2048
Max. Overhead	21%	10%	16%

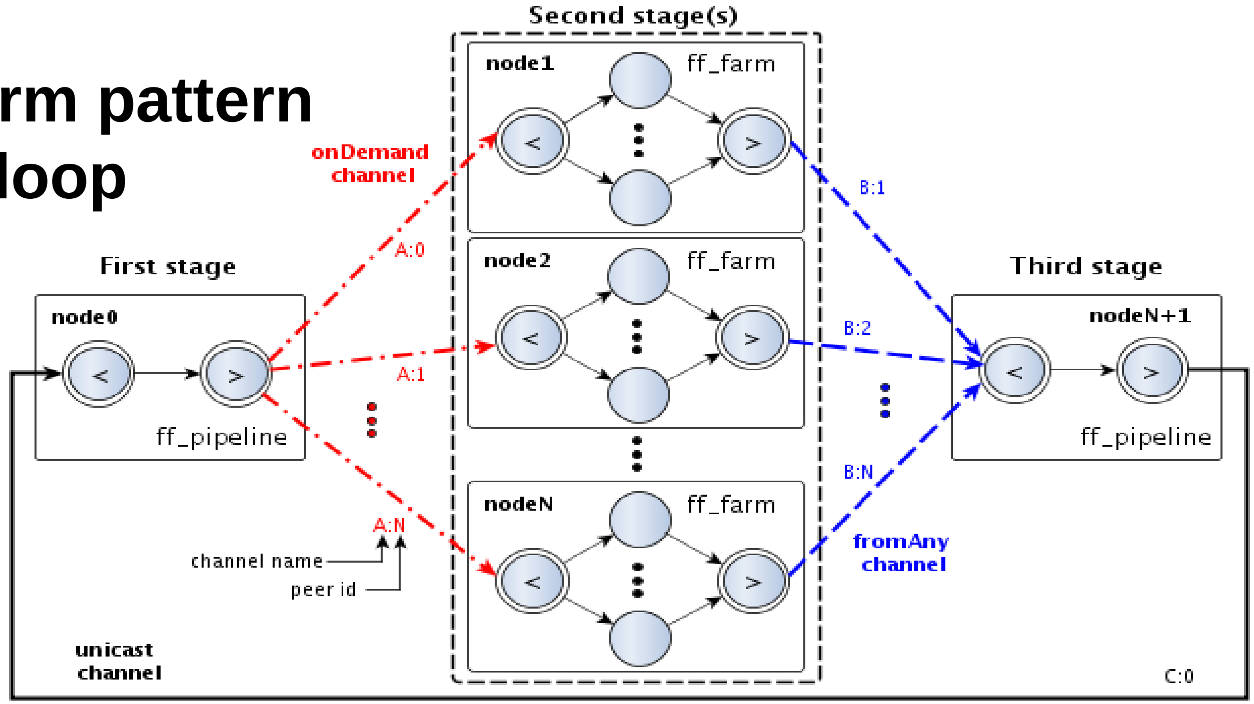
Amazon EC2 public cloud

- Distributed virtual environment : Amazon EC2
    - 7 Linux Ubuntu x86_64 virtual machines
    - Intel CPU E-2670 @2.6GHz with 20MB L3 cache



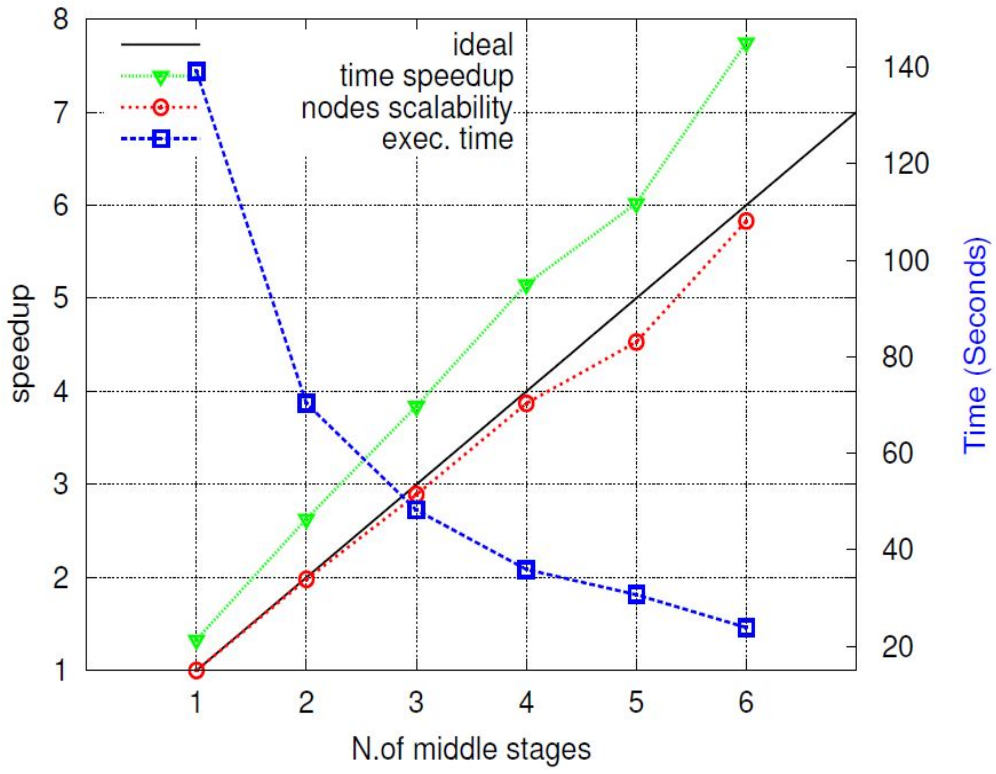
Distributed test

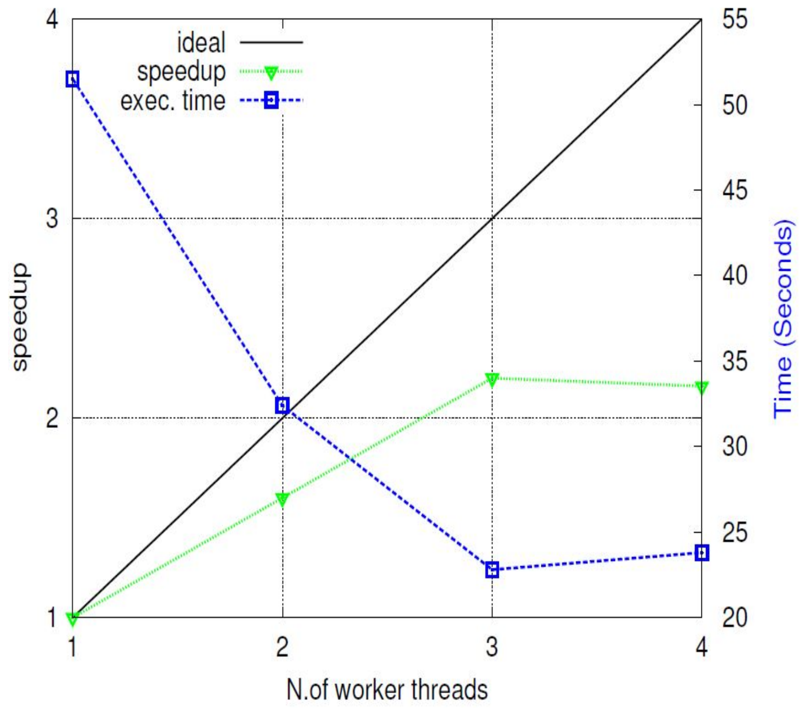
Farm of Farm pattern with outer loop



- The 1st stage produces a stream of matrices (double elements)
- The middle stage computes the square of each input
  - It is internally parallel (task-farm skeleton)
- The last stage collects and sends back results to the 1st stage.

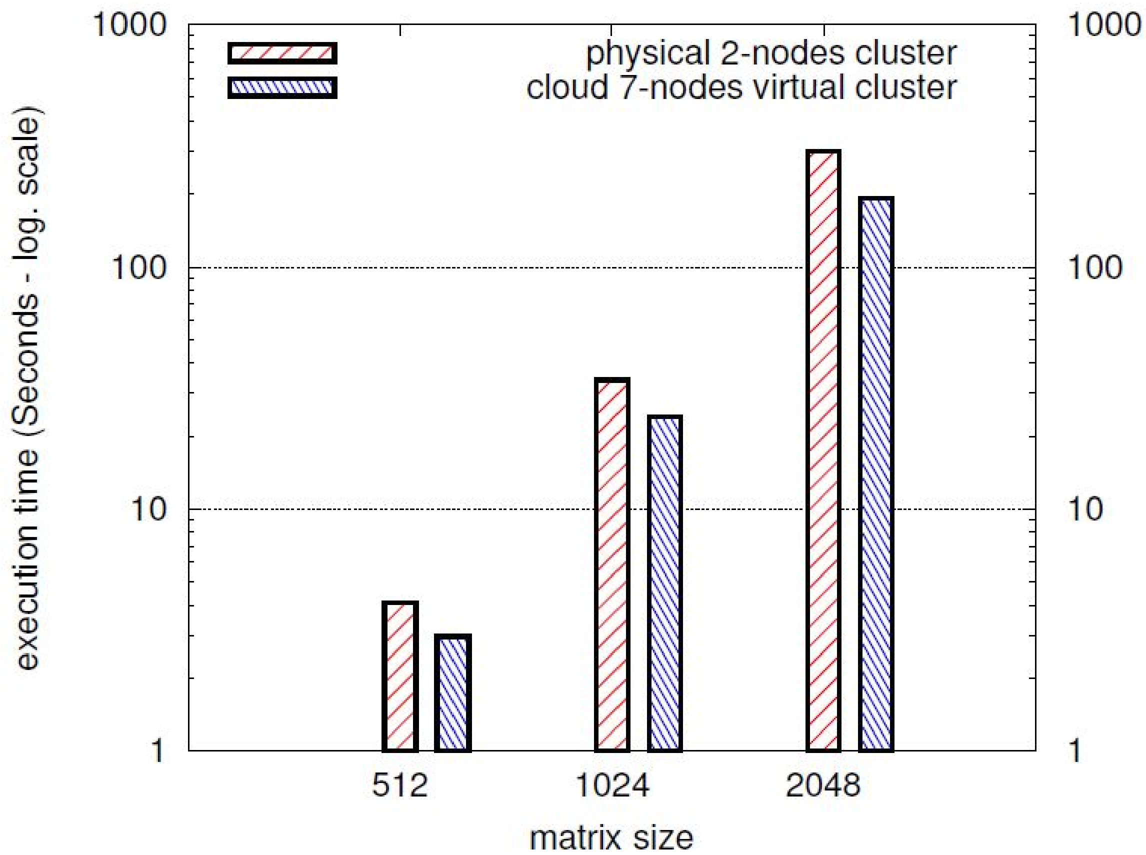
Performance





- Matrix size 1024x1024 double elements (8MB)
- For the graph in the right-hand-side we used 6 VMs

Performance



- Physical cluster: 2 nodes each one with 2 Intel Xeon CPUs E5-2650 @2.0GHz 8-cores 16-context with 20MB L3 cache (32 cores total)
- The 2 nodes are connected using Infiniband Card (40Gb/s)

Conclusions

- Tested FastFlow on virtual multi-core machine and virtualized public cloud infrastructure
- Virtualized execution is costly, but overhead is predictable and bounded
  - 2-30% for our benchmark on KVM-based virtualization
- FastFlow can be smoothly deployed on public cloud obtaining good performance
- The strength of the approach lies on the structured parallel programming methodology adopted
- FastFlow is the first parallel skeleton programming environment to be efficiently executed on the cloud

Further Reading

- Campa S, Danelutto M, Goli M, Gonzalez-Velez H, Popescu AM, Torquati M. Parallel patterns for heterogeneous CPU/GPU architectures: Structured parallelism from cluster to cloud. Future Generation Computer Systems. 2014; 37:354–366. doi:10.1016/j.future.2013.12.038

- Goli M, Gonzalez-Velez H. N-body Computations Using Skeletal Frameworks on Multicore CPU/graphics Processing Unit Architectures: an Empirical Performance Evaluation. Concurrency and Computation–Practice & Experience. 2014; 26(4):972–986. doi:10.1002/cpe.3076

- Boob S, Gonzalez-Velez H, Popescu A. Automated Instantiation of Heterogeneous FastFlow CPU/GPU Parallel Pattern Applications in Clouds. In: PDP’14. Torino: IEEE; 2014. p. 162–169.

Questions, Comments or Suggestions?