High-Performance Computing: Gossip, Lies, & Secrets

Horacio González-Vélez
Cloud Competency Centre, NCI
E: horacio@ncirl.ie
horaciogv

22-Sep-16

Agenda

- speaker
- background
- programming
- conclusions

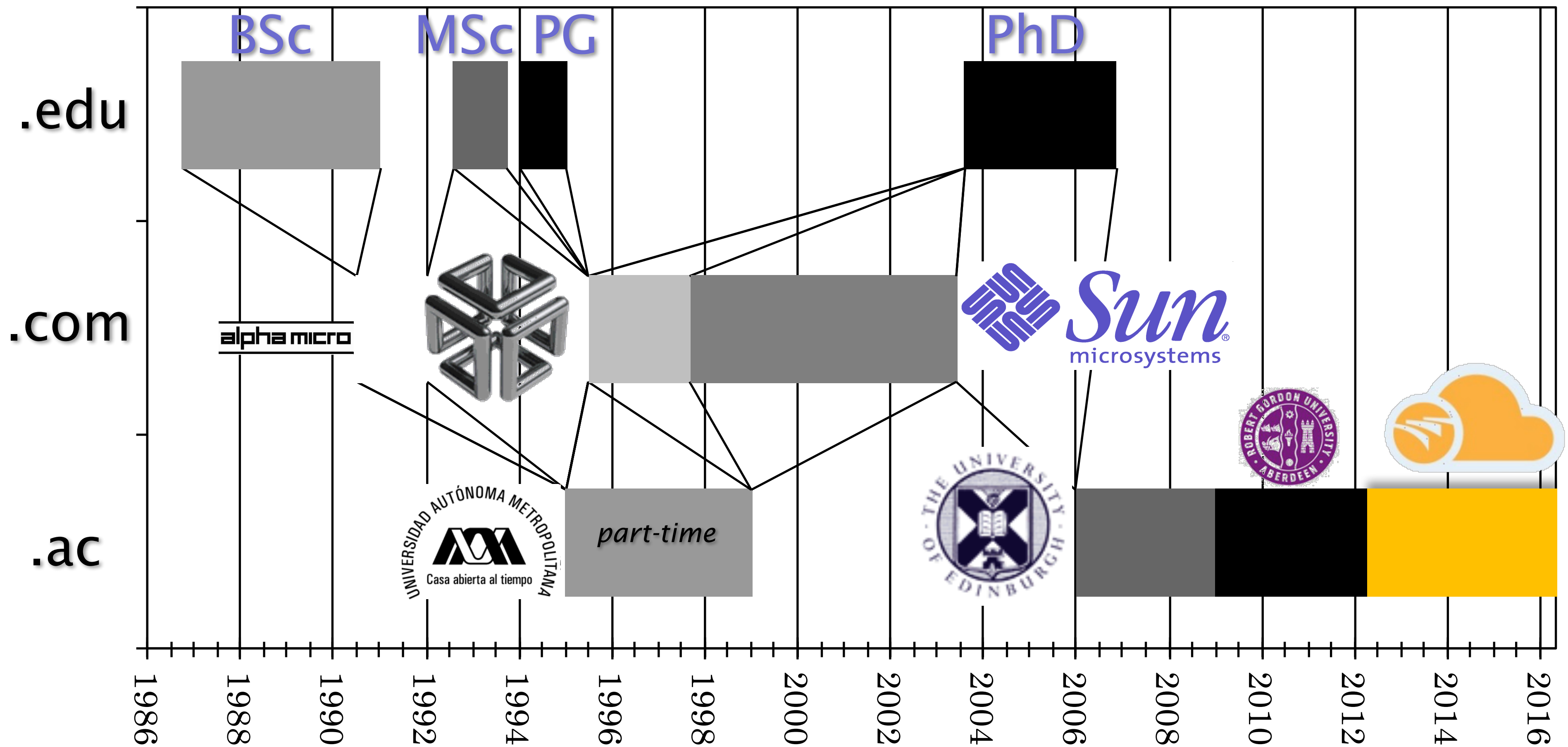
speaker

spanish 101

```
</begin>
let h be silent
let c be s
if President_Obama  is
Irish
   then
    let Horacio be O'Rassio
</end>
```

cHiPSet

career



25+ years @ HPC & parallel computing

research

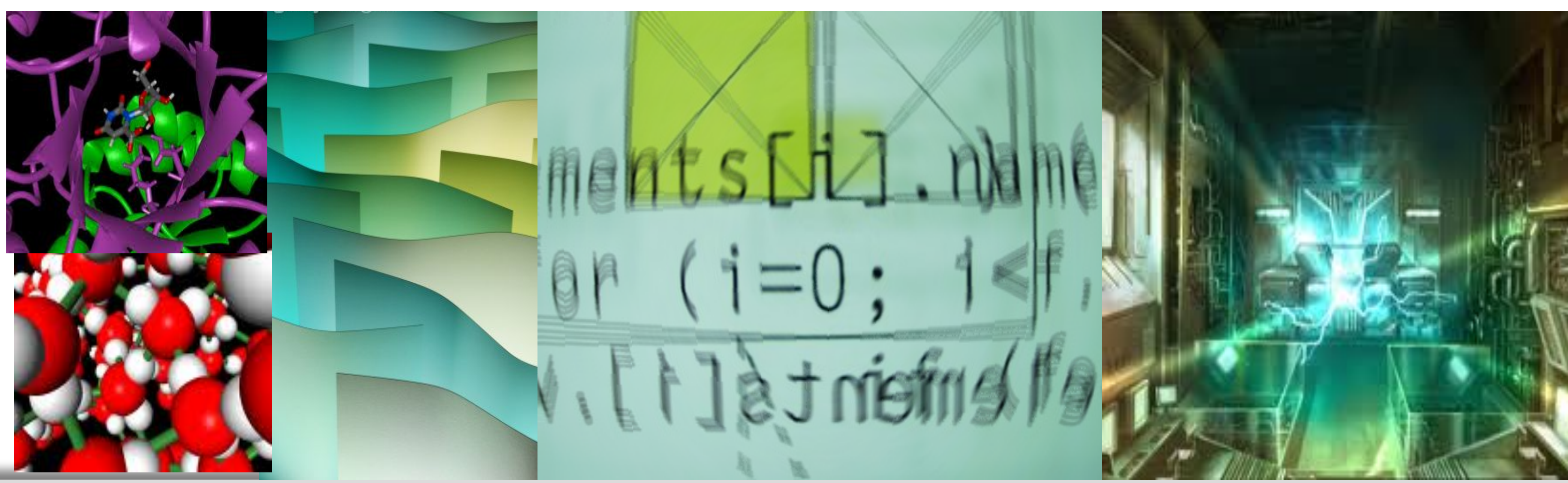
integration of

computational problems &

parallel patterns

to adaptively improve overall

resource utilisation



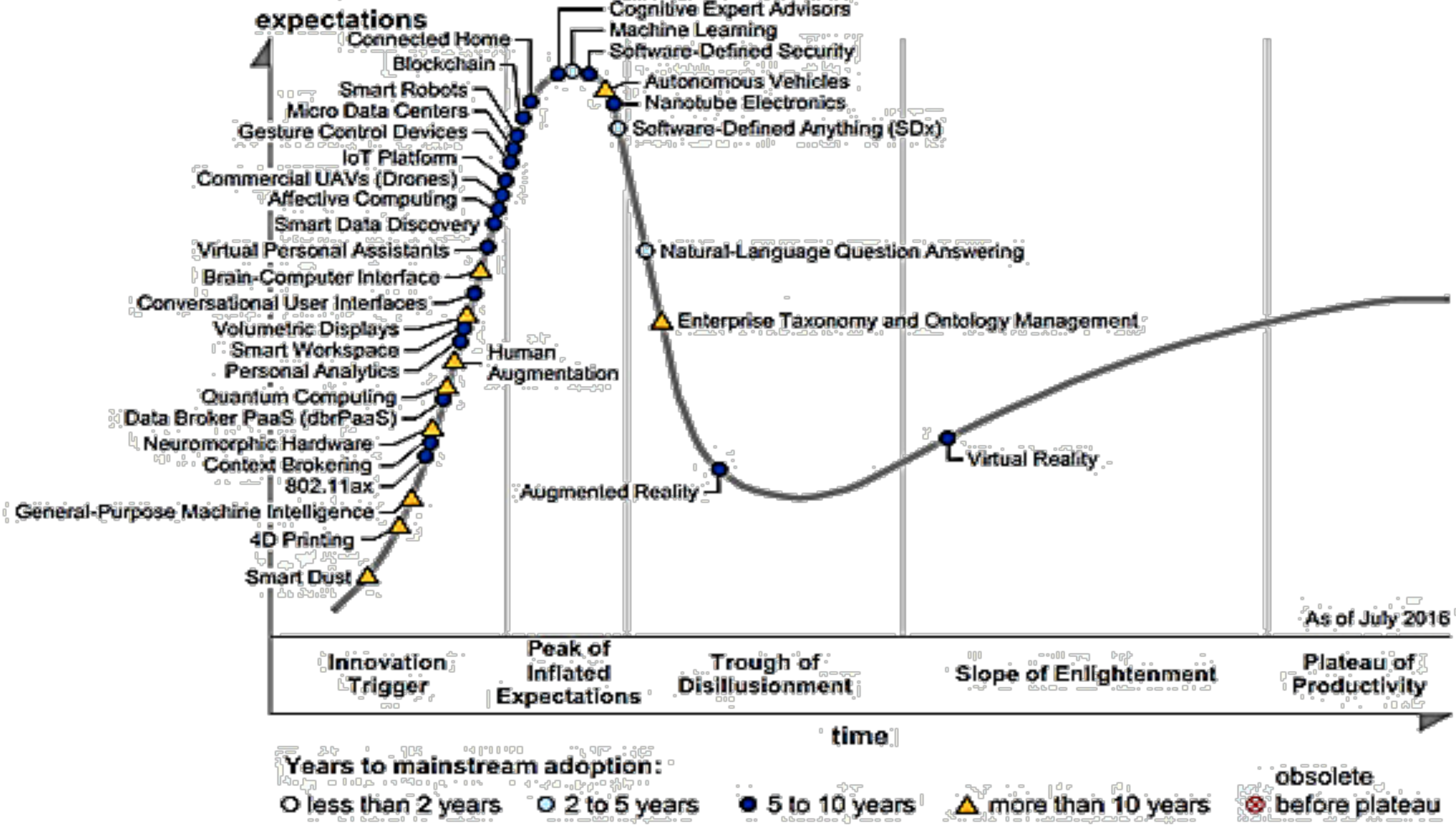
background

HPC Evolution

- HPC has moved from centralised supers through P2P, minis, clusters, and grids to clouds over last 40 years
- R/D efforts on HPC, clusters, Grids, P2P, and virtual machines has laid the foundation of cloud computing
- Location of computing infrastructure in areas with lower costs in hardware, **software**, **datasets**, space, and **power** requirements – moving from desktop computing to datacenter-based clouds

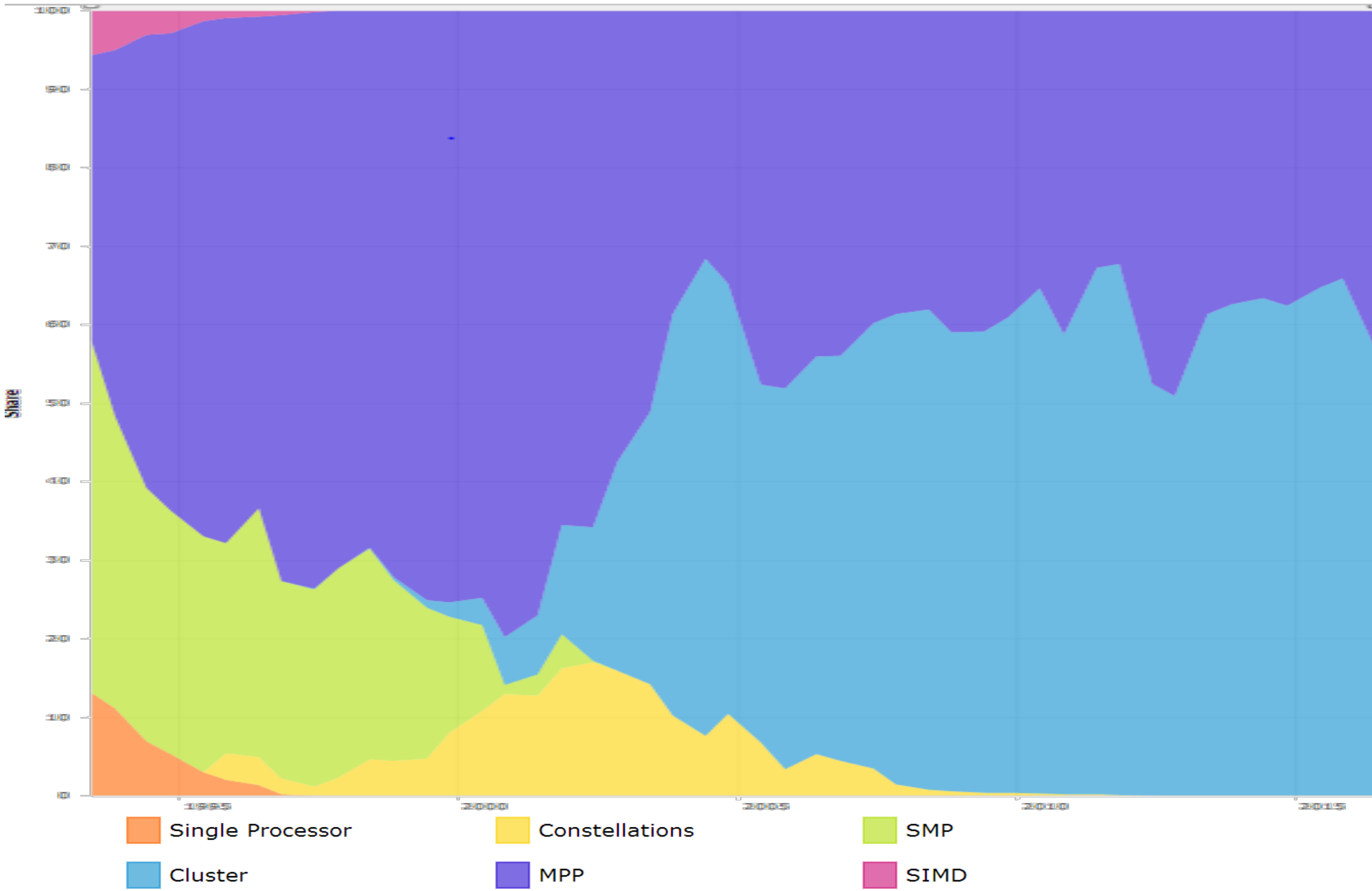
Source: K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing,* Morgan Kaufmann, 2012.

Gartner's 2016 hype cycle



Source: Gartner (July 2016)

top 500


Share
Single Processor
Constellations
SMP
Cluster
MPP
SIMD

flops dance

Source: Wikipedia

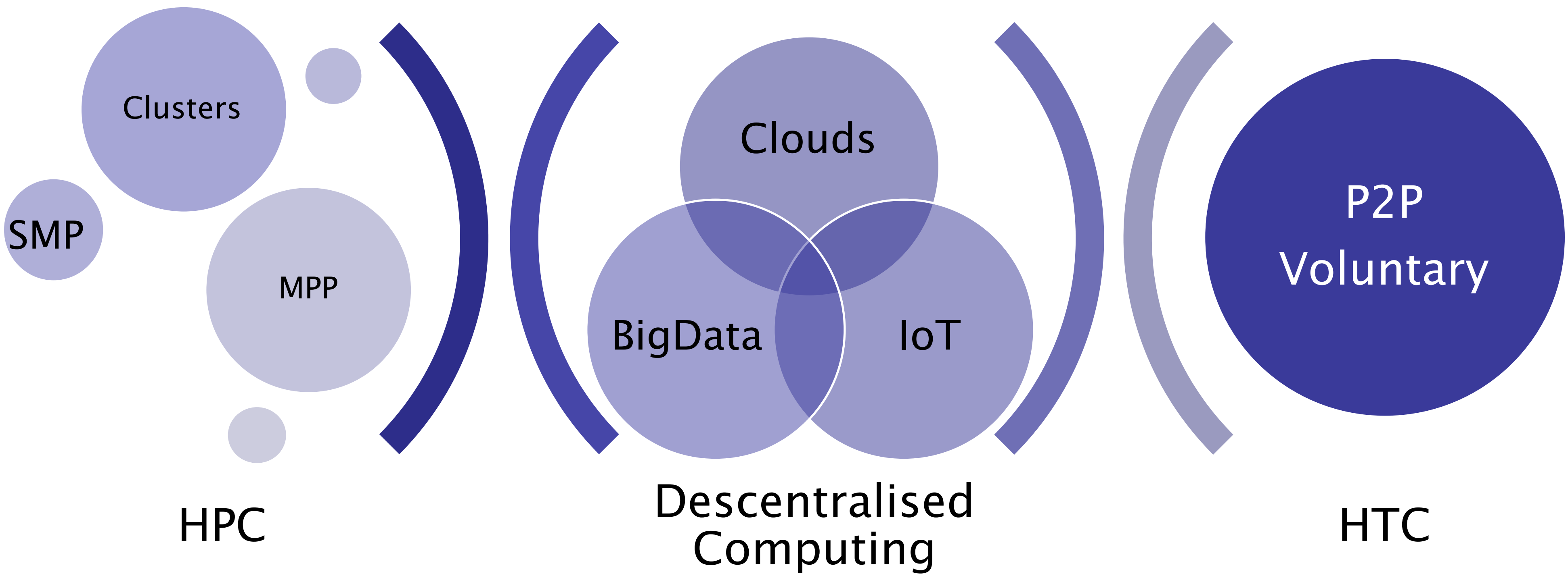
1x Flops,
5x mem,
.001x cost

1976: Cray 1
160MFlops, 1MWords, $1M

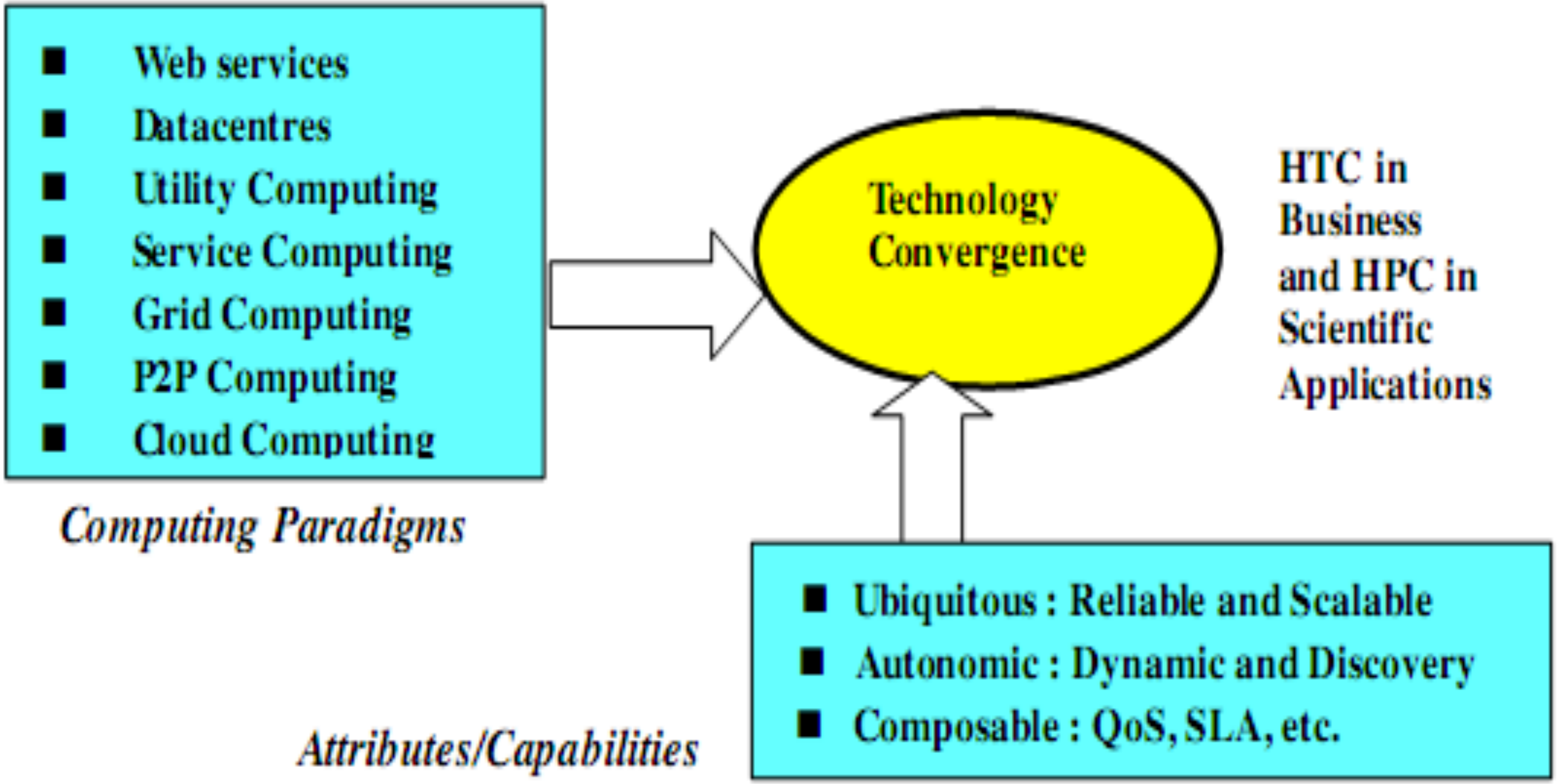
2016: iPhone 7
2.23GFlops*, 256GB, $1K

cHiPSet

HPC vs HTC

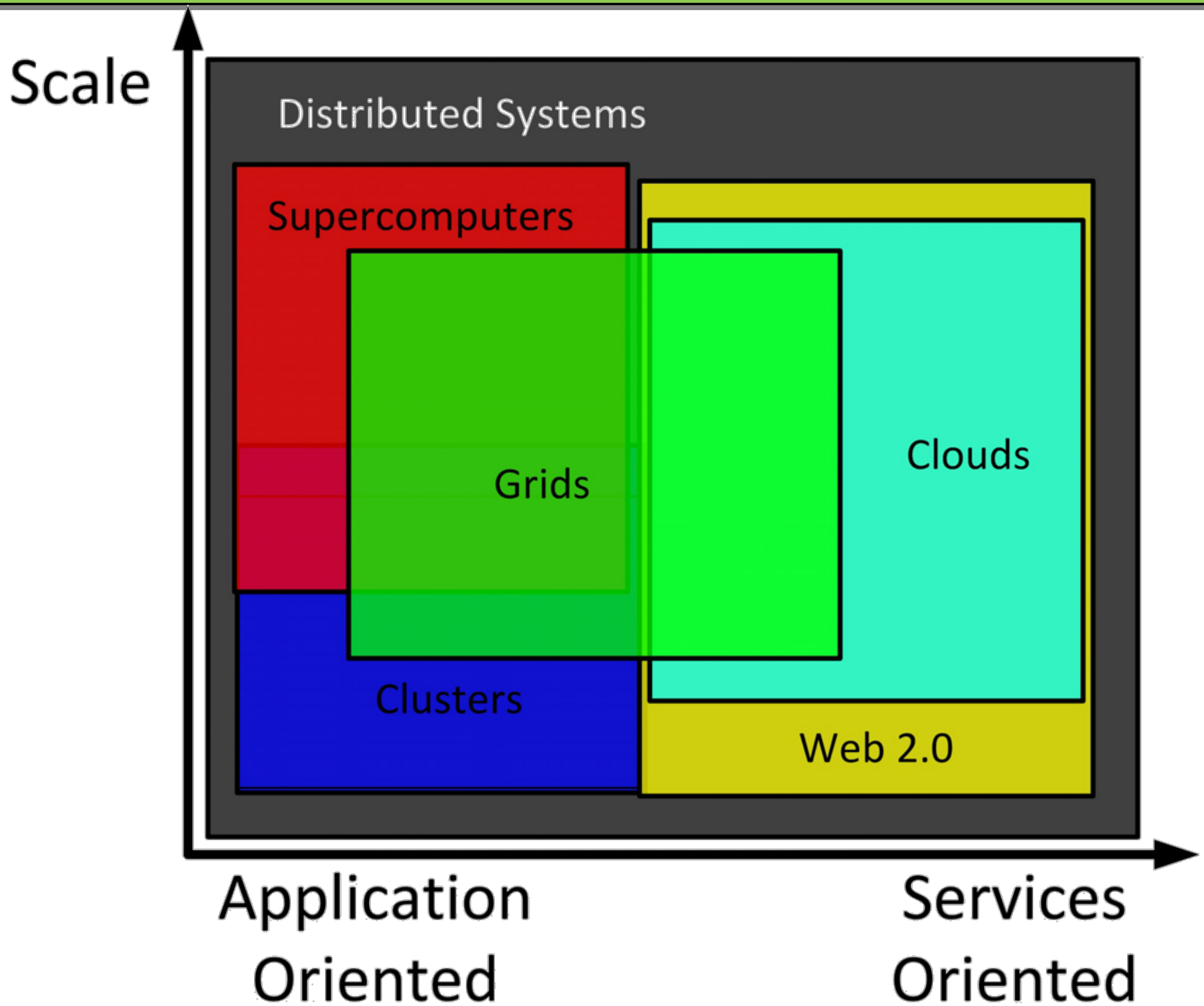


HPC.ac HTC.biz



Source: Raj Buyya, University of Melbourne, 2011

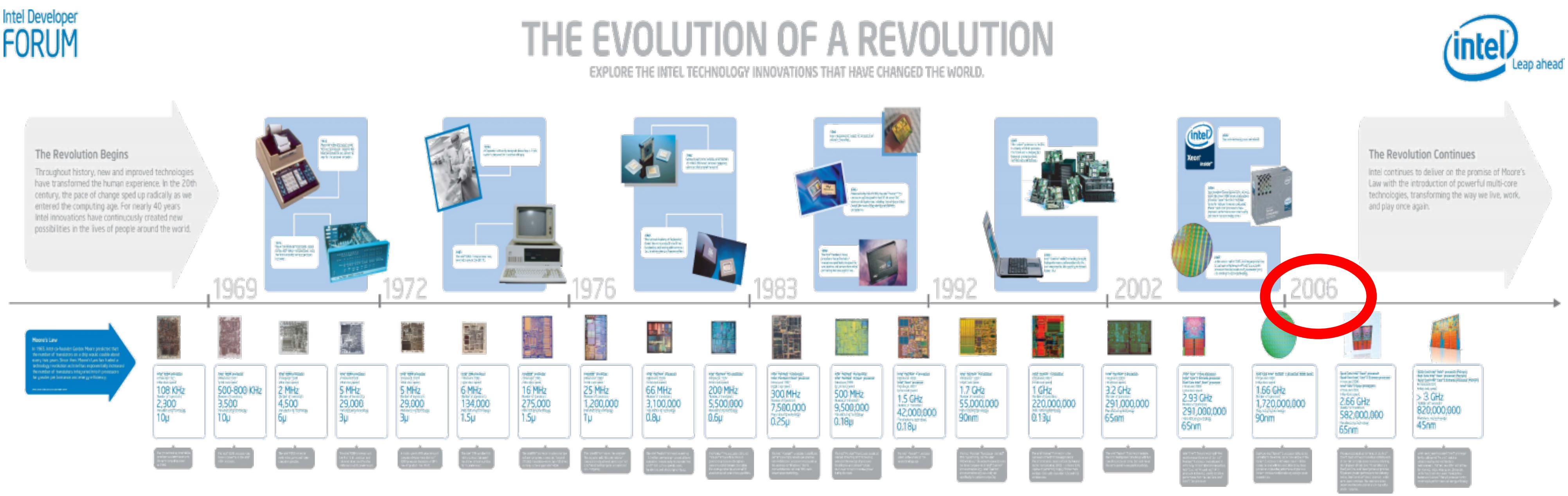
when to use what



Foster et al. "Cloud Computing and Grid Computing 360-Degree Compared," GCE '08 , Nov. 2008 doi: 10.1109/GCE.2008.4738445

cHiPSet

incomplete evolution

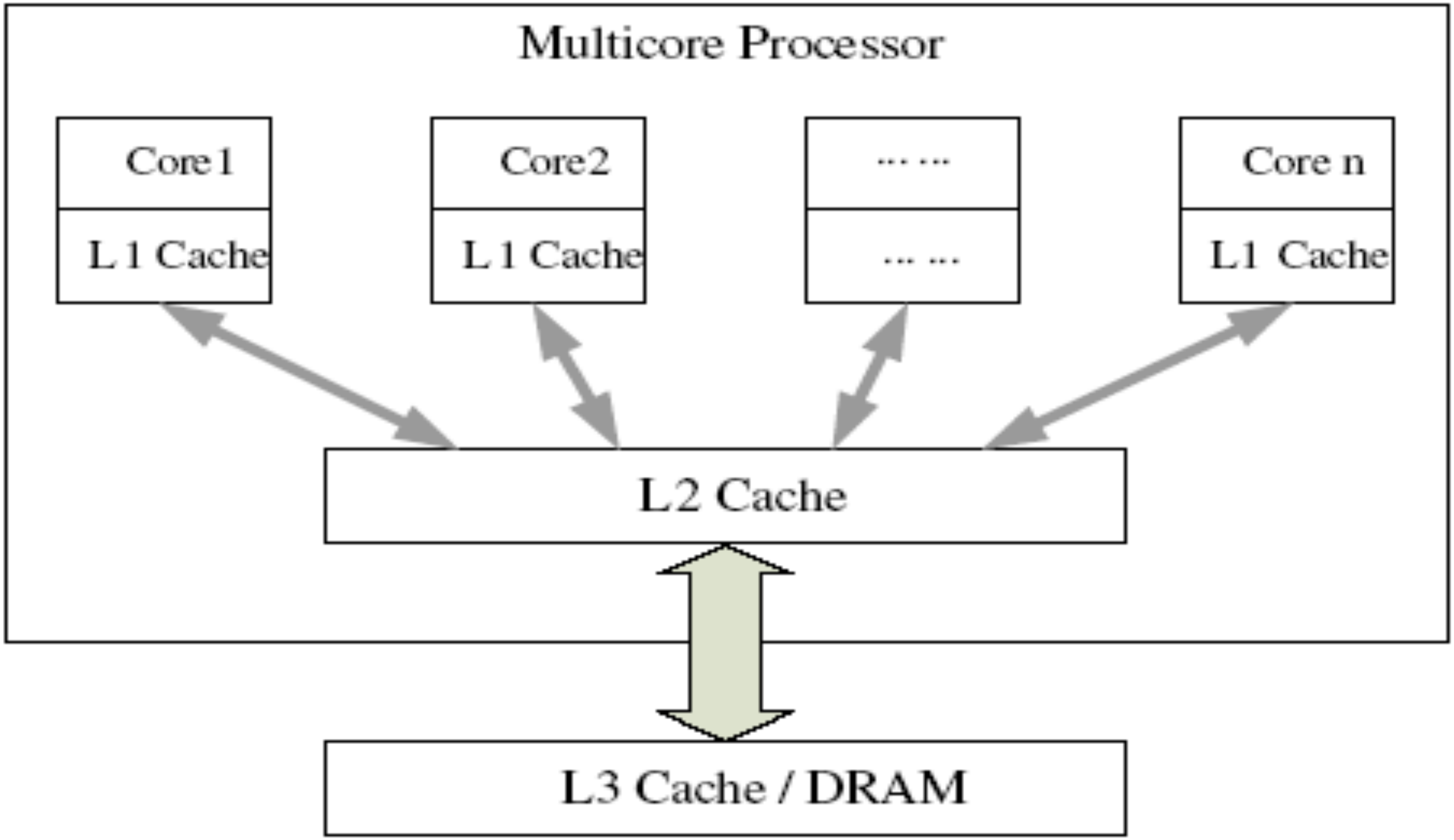


http://download.intel.com/pressroom/kits/IntelProcessorHistory.pdf

Intel only kept its **Evolution of a Revolution** chart up to 2006

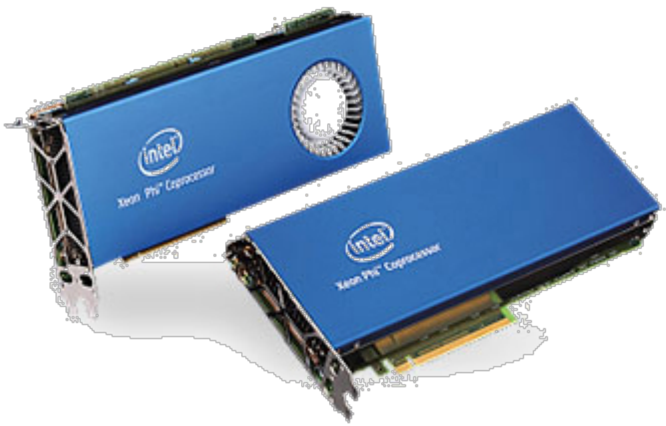
Why?

multicores



Multicore Processor
Core 1
L 1 Cache
Core2
L 1 Cache
... ...
... ...
Core n
L1 Cache
L2 Cache
L3 Cache / DRAM

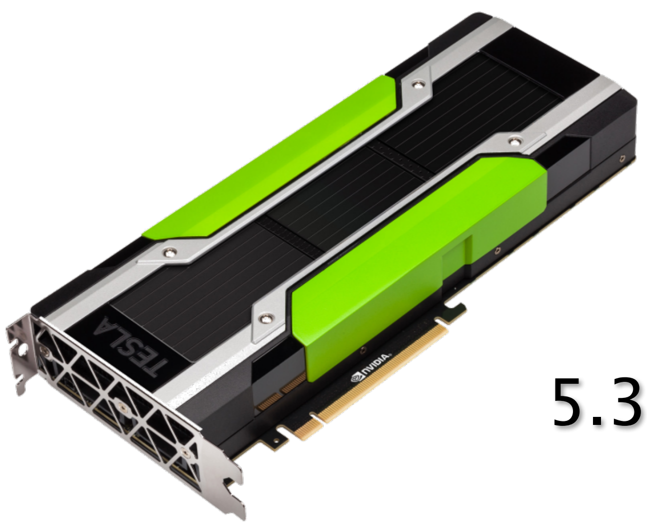
accelerators to the rescue



Intel Xeon Phi

72 cores, 288 threads, 3+TFlops DP

Cori @ NERSC with 9300 Phi

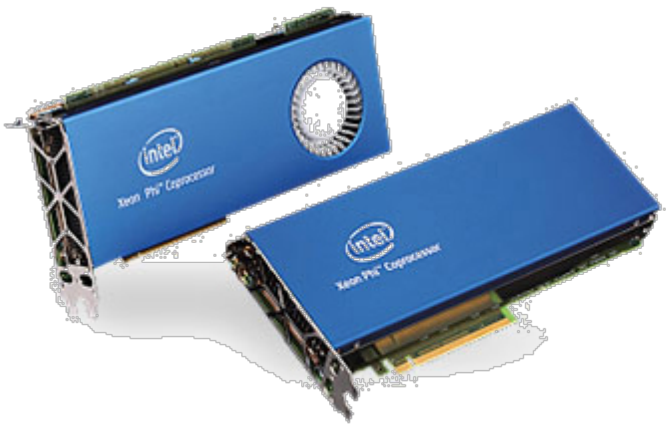


NVIDIA Tesla P100

5.3 TFlops DP 64-bit, 3584 cores, 300W

cHiPSet

sponsored



Intel Xeon Phi

NVIDIA Tesla

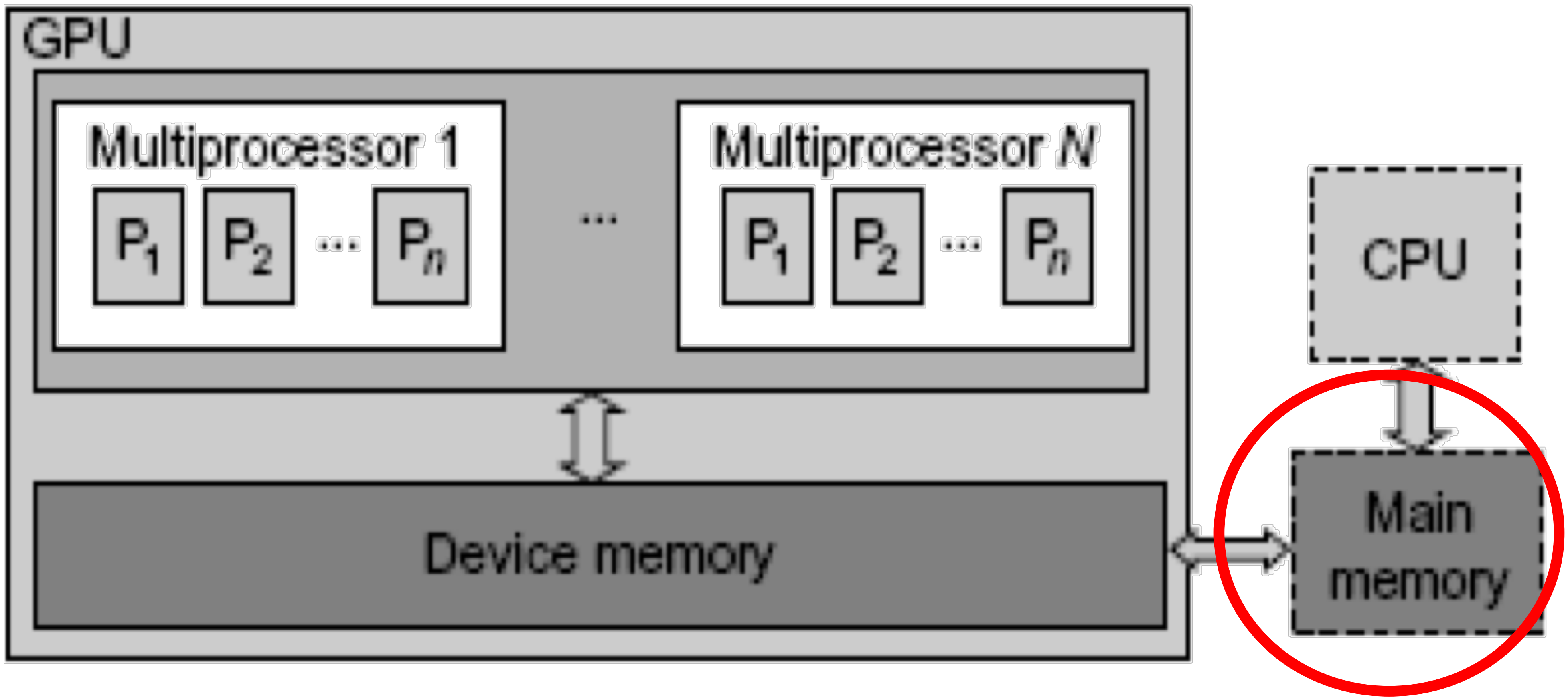
for cHiPSet STSMs

at

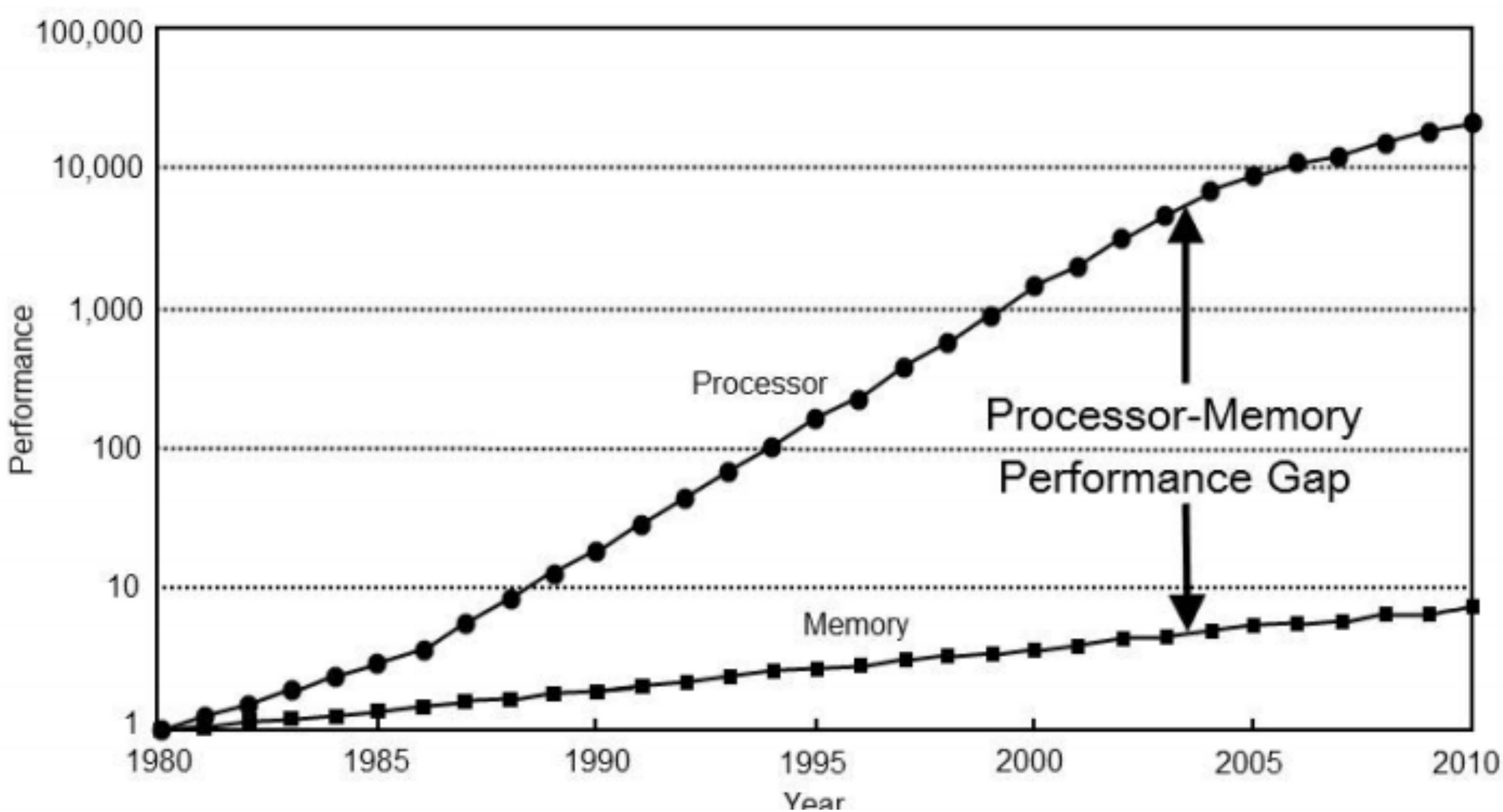
National College of Ireland

Cloud Competency Centre

memory bottleneck



cpu-memory gap



challenge

"Ultimately, developers should start thinking about ***tens, hundreds, and thousands*** of cores ***now*** in their algorithmic development and deployment pipeline."
Anwar Ghuloum, Principal Engineer, Intel Microprocessor Technology Lab

"The dilemma is that a ***large percentage*** of mission-critical enterprise applications will ***not*** ``automagically'' run ***faster*** on multi-core servers. In fact, many will actually *run **slower***. We must make it as easy as possible for applications programmers to exploit the latest developments in multi-core/many-core architectures, while still making it easy to target future (and perhaps unanticipated) hardware developments."
Patrick Leonard, Vice President for Product Development
Rogue Wave Software

programming

Sad but true…

There are two things in life you cannot buy (get enough of):

LOVE

&

SCALABILITY

typical approaches

- Applications Programmers *= Systems Programmers*
  - Insufficient assistance with abstraction
- Tough to scale, unless the problem is simple
- Difficult to change fundamentals
  - Scheduling, Task structure, Migration
- Abstractions NEEDED

multi-topic

The 2012 ACM Computing Classification System -

- [A] General Literature; [B] Hardware
- [C: Computer Systems Organisation]
  - C.1 Architectures
    - C.2.2 Parallel Architectures
    - C.2.3 Distributed architectures
- [D] Networks
- [E: Software and its Engineering]
  - E.1 Software Organisation and properties
    - E.1.3 Extra Functional Properties: Interoperability, performance, reliability, usability
  - E.2 Software Notations and Tools
    - E.2.1 General Programming Languages: Language Features- Patterns || Concurrent Programming Structures
- [F] Data; [G] Theory of Computation; [H] Mathematics of Computing; [I] Information Systems; [J] Security and Privacy; [J] HCI;
- [K: Computing Methodologies]
  - K.1 Parallel Computing Methodologies
- [L] Applied Computing; [M] Social and professional topics

cHiPSet

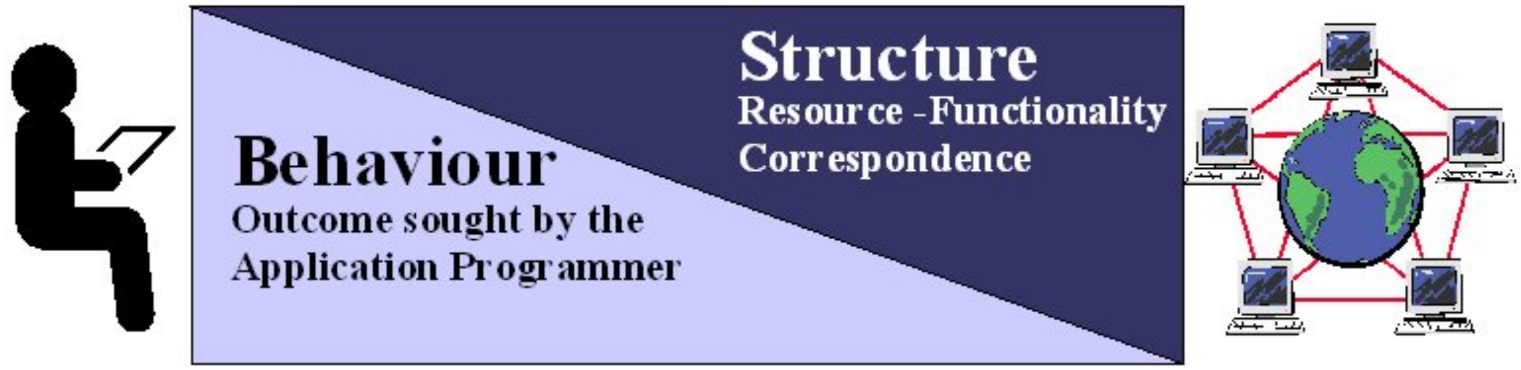
issues

- We can muddle through on 2-8 cores
  - maybe even 16
  - modified sequential code may work
  - multiple programs to soak up cores
  - BUT larger systems are *much* more challenging

- "Think parallel"
  - New *high-level* programming constructs
  - Decouple Computation from Coordination

algorithmic skeletons

- Higher-Order Functions
- Abstract **Patterns** of Parallel Computation, Communication, and Interaction
- Decouple **Behaviour** (Computation) from **Structure** (Coordination)



M Cole: Algorithmic skeletons: structured management of parallel computation. MIT Press, 1991.

cHiPSet

classification

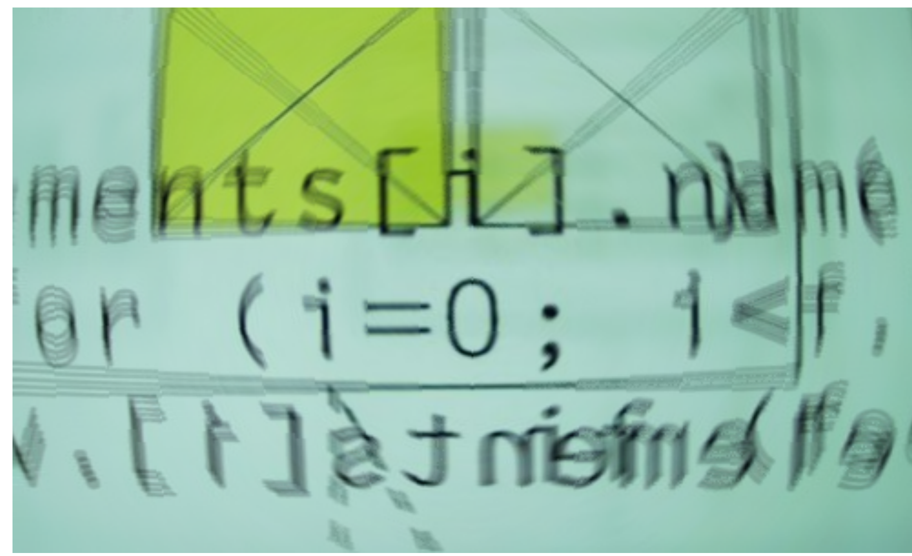
Skeleton	Scope	Example
Data-Parallel	Data Structures	Scan, Map, Broadcast, Reduce, Gather, Scatter,
Task-Parallel	Tasks	Farm, Pipeline, Seq, …
Resolution	Family of Problems	Div &Conq, Br & Bnd, Dyn Prog, Heuristic Opt,

Gonzalez-Velez H, Leyton M. A Survey of Algorithmic Skeleton Frameworks: High-Level Structured Parallel Programming Enablers. *Software: Practice and Experience*. 2010 Dec;40(12):1135-1160. [http] .

cHiPSet

structured parallelism

- Based on skeletons, **Structured Parallelism** provides:
  - Top-down design and construction
  - Well-defined control structures
  - Fixed scope of data structures



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

Open Question

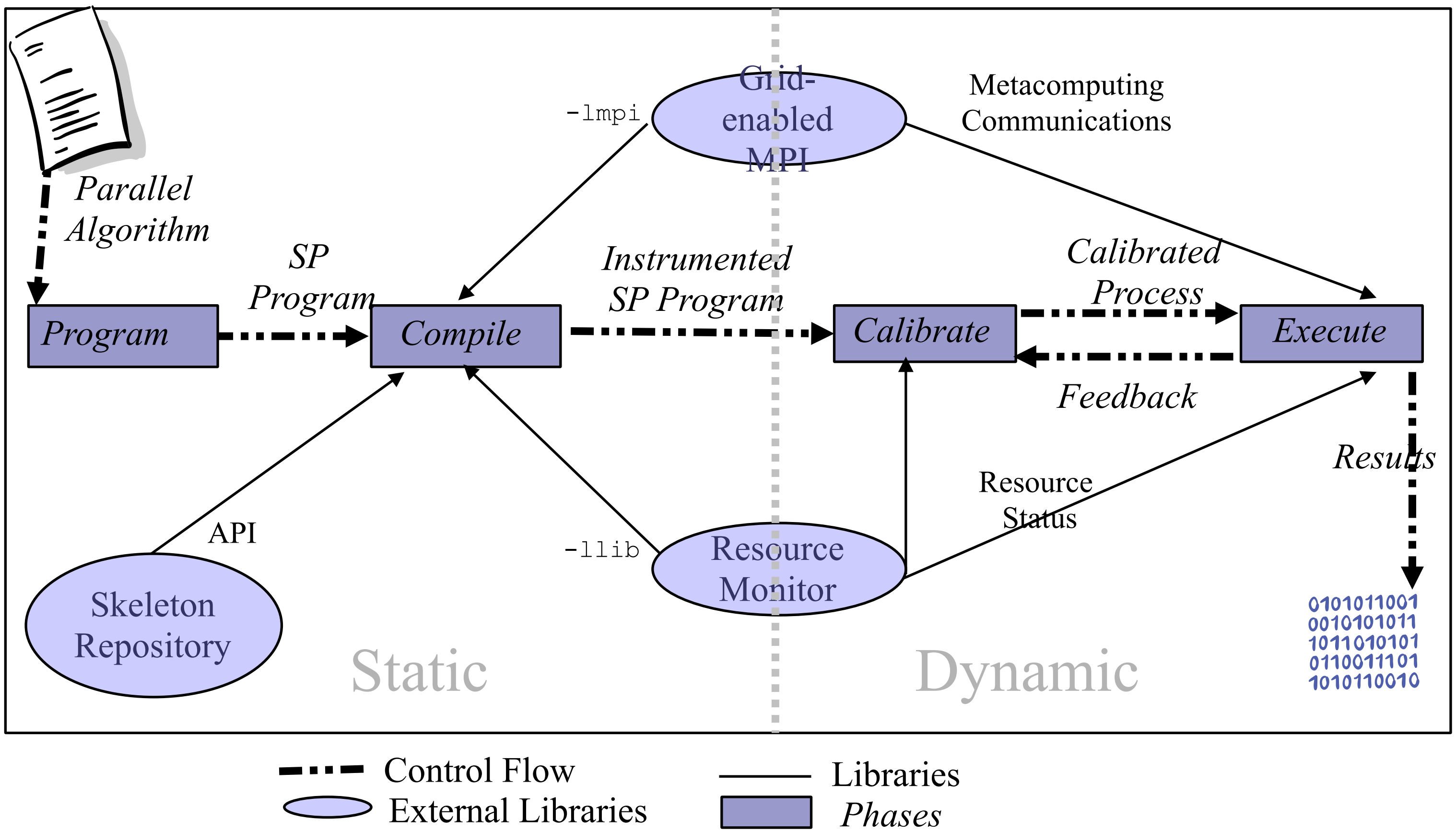
Can the skeletons **improve** the Performance of Parallel Applications executing in a **non-dedicated** heterogeneous System?

cHiPSet

Motivation

- Compilers are Static
- Run-time Optimisers are too General
- Skeletons have Structured, Predictable Behaviour for a given Program
- **Hypothesis: A Skeletal Program should be able to Adapt to Dynamic Resource Conditions over time using its Structural Forecasting Information**

Methodology



Phases

- **Program**: Select algorithmic skeleton and parameterises the API
- **Compile**: Link with required libraries
- **Calibrate**: Execute worker/stage function on input subset, extrapolate node fitness, and rank nodes
- **Execute:** Monitor grid resource usage and adapt workload accordingly

phases

- **Program**: Select algorithmic skeleton and parameterises the API
- **Compile**: Link with required libraries
- **Calibrate**: Execute worker/stage function on input subset, extrapolate node fitness, and rank nodes
- **Execute:** Monitor grid resource usage and adapt workload accordingly

phases

- **Program**: Select algorithmic skeleton and parameterises the API
- **Compile**: Link with required libraries
- **Calibrate**: Execute worker/stage function on input subset, extrapolate node fitness, and rank nodes
- **Execute:** Monitor grid resource usage and adapt workload accordingly

phases

- **Program**: Select algorithmic skeleton and parameterises the API
- **Compile**: Link with required libraries
- **Calibrate**: Execute worker/stage function on input subset, extrapolate node fitness, and rank nodes
- **Execute:** Monitor grid resource usage and adapt workload accordingly

phases

- **Program**: Select algorithmic skeleton and parameterises the API
- **Compile**: Link with required libraries
- **Calibrate**: Execute worker/stage function on input subset, extrapolate node fitness, and rank nodes
- **Execute:** Monitor grid resource usage and adapt workload accordingly

CONTRIBUTION

cHiPSet

implementation

- C APIs + MPI
- 2 Skeletons but GRASP is **NOT** restricted

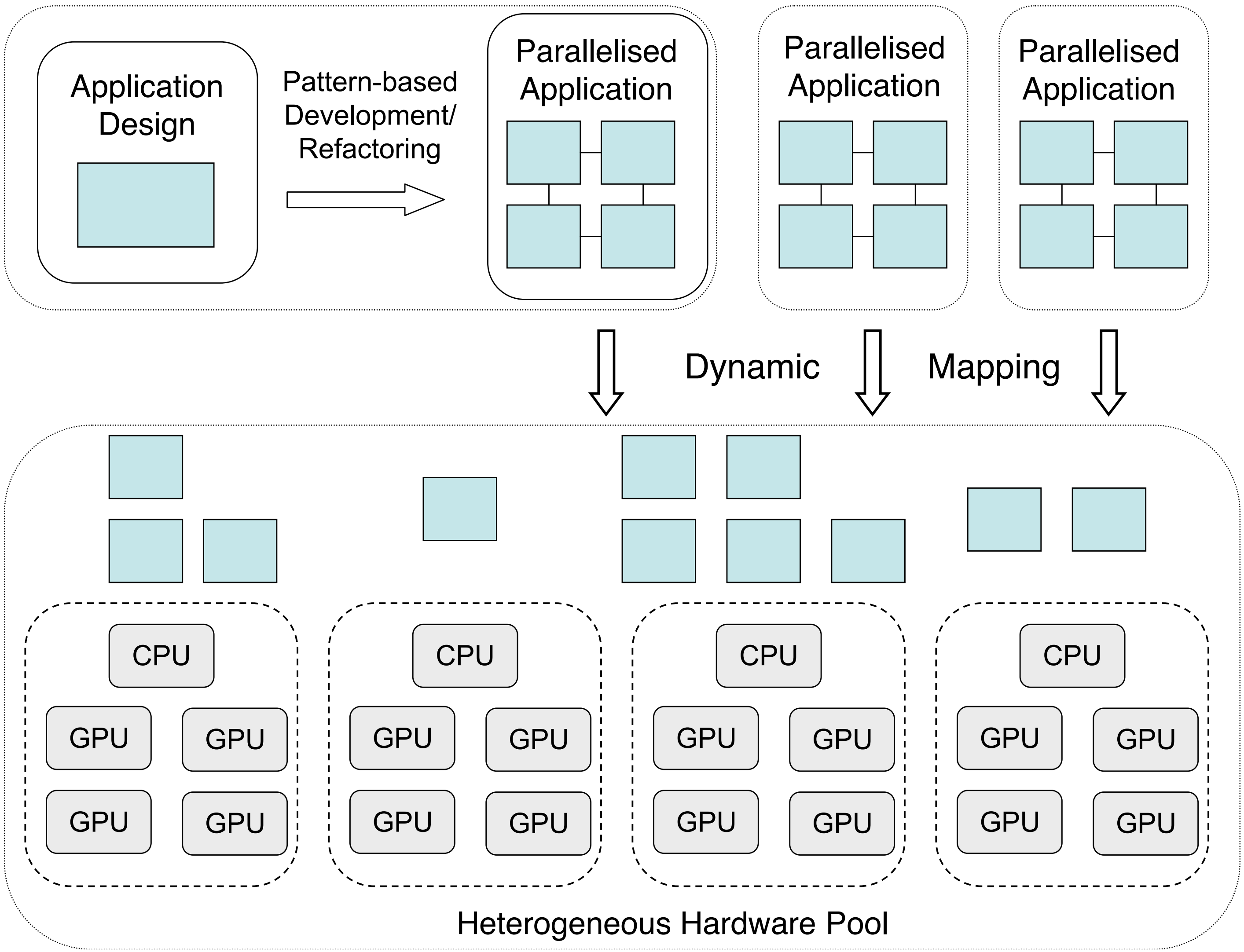
Algorithmic Skeleton	Workload Type	Computation Type	Application Employed
Task Farm	Disjunct	Embarrassingly-parallel	Computational Biology Parameter Sweep
Pipeline	Precedence relations	Pipelined	Whetstones Benchmark Function

- Individual Tasks with Similar Complexity
- **2006-2010 (then)**

'10s: ParaPhrase

- 3.5 Year targeted research project (FP7 STReP)
  - Runs from 1/10/11 to 31/3/15
  - Funded by the European Commission
- 13 partners from 8 countries
  - Austria, Germany, Ireland, Israel, Italy,
  - Hungary and Poland
- € 4.2M

patterns multicore / gpu

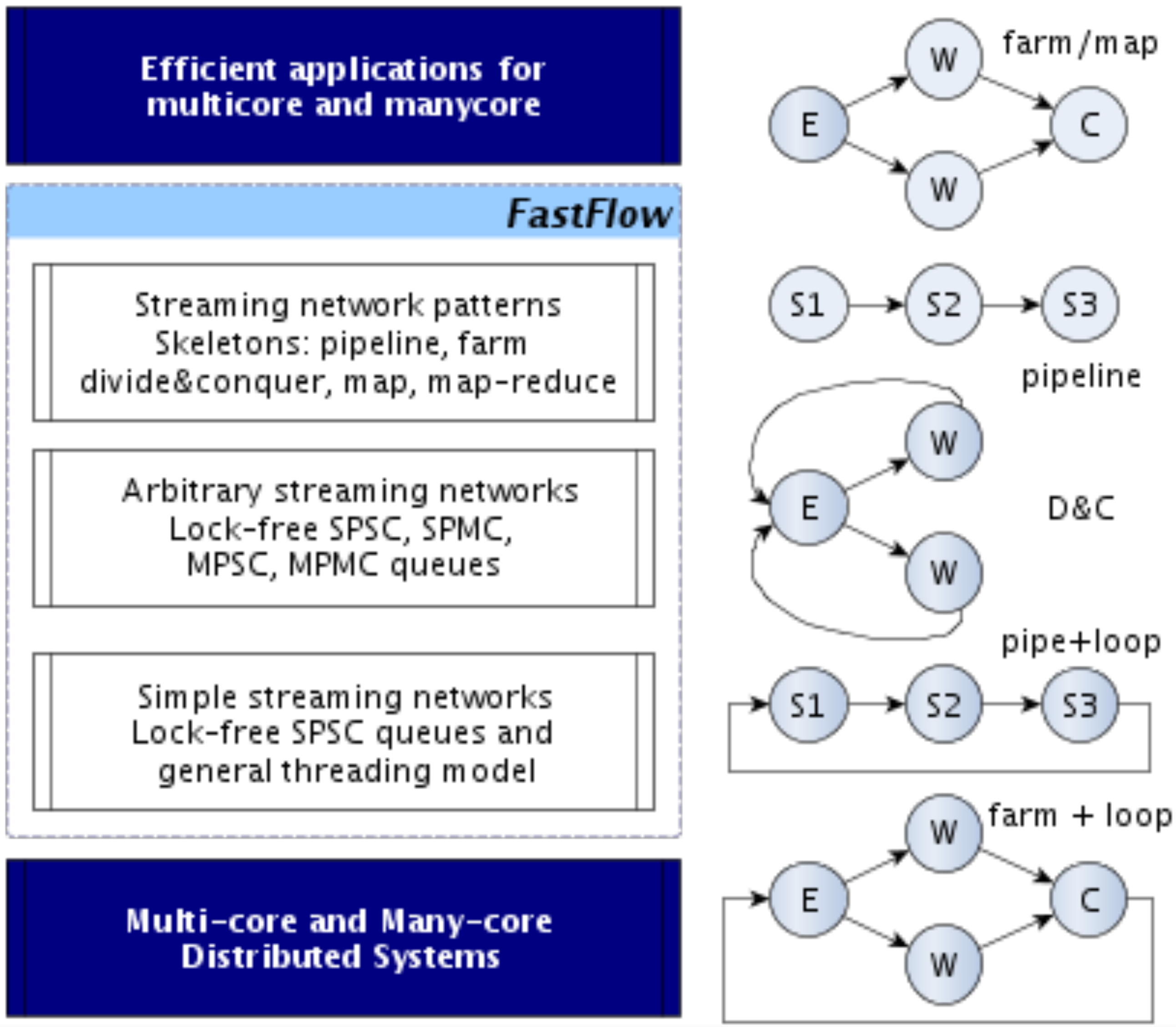


fastflow

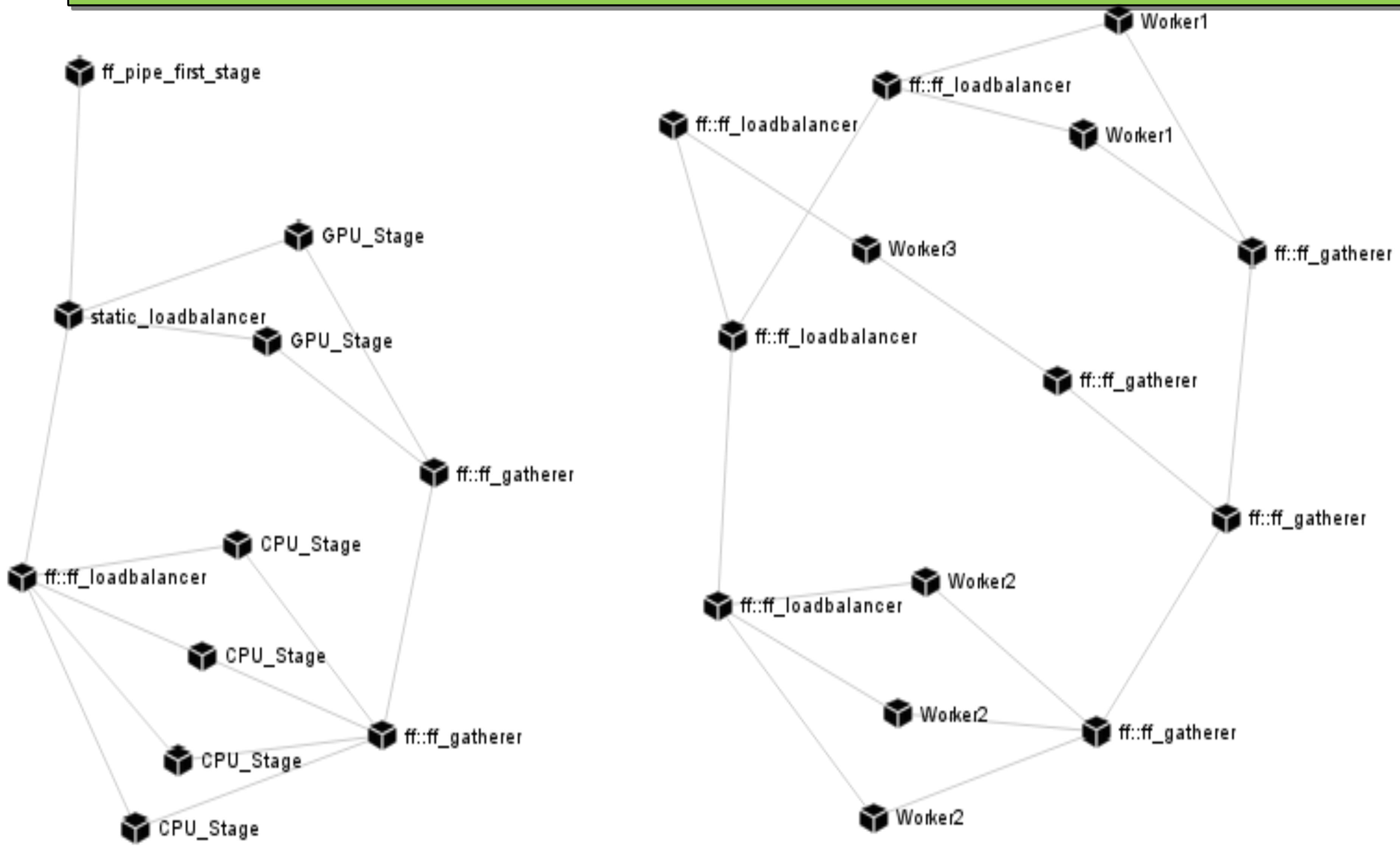
- Structured parallel programming framework
- FastFlow: Skeletons = C++ classes & templates (via Pthreads).
- Target: Multi-core CPU, Dist Sys, GPU
- Stream parallel patterns: pipeline, task-farm, loopback
  - Ongoing work for map and map-reduce skeletons on multi-core
- Task-offloading on Tile64 and GPUs
- ParaPhrase Programming Framework. Open Source (developers in cHiPSet WG2)

http://calvados.di.unipi.it/

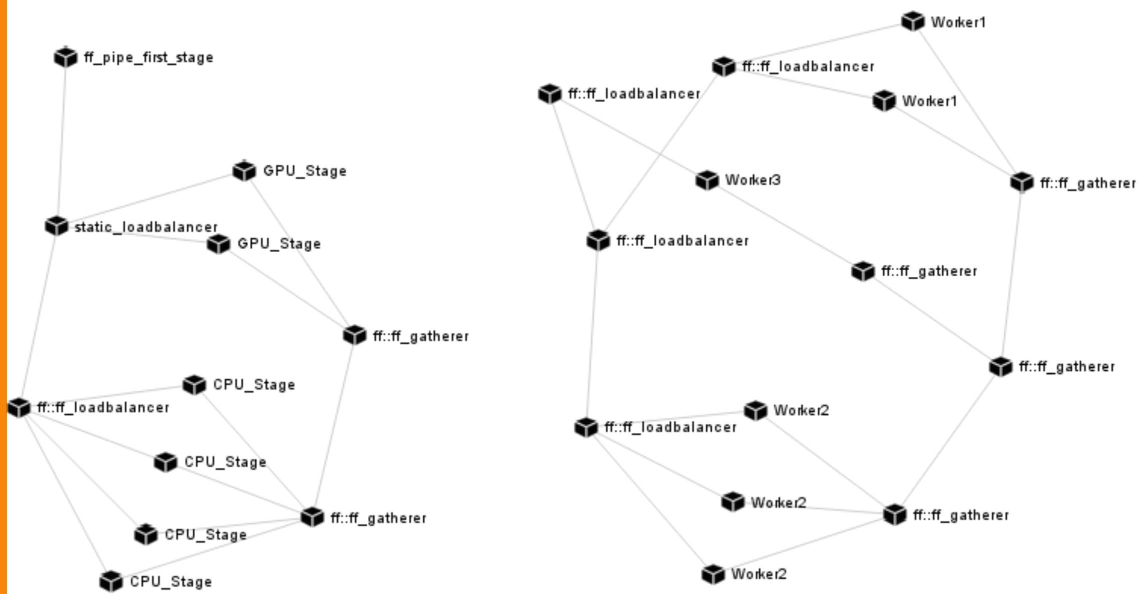
concepts

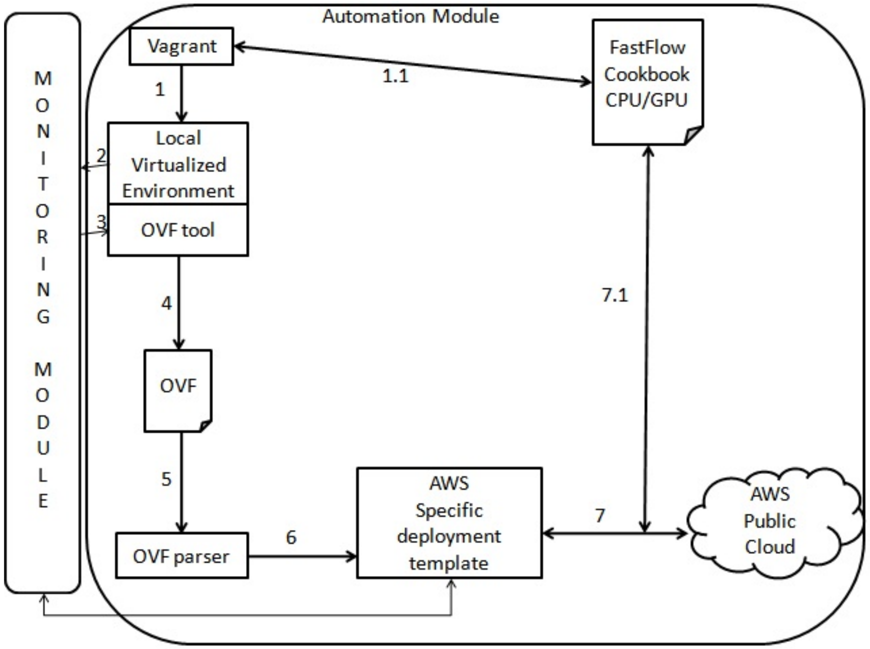

Efficient applications for multicore and manycore
FastFlow
Streaming network patterns
Skeletons: pipeline, farm
divide&conquer, map, map-reduce
Arbitrary streaming networks
Lock-free SPSC, SPMC,
MPSC, MPMC queues
Simple streaming networks
Lock-free SPSC queues and
general threading model
Multi-core and Many-core
Distributed Systems
farm/map
E
W
W
C
S1
S2
S3
pipeline
D&C
pipe+loop
farm + loop

visualisation



Elastic deployment





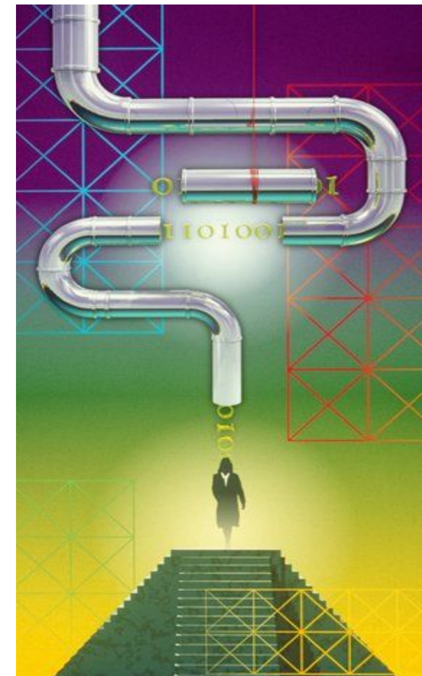
conclusions

cHiPSet

key findings

- Structure-based Resource-Awareness improves the Performance of Skeletal Programs in Heterogeneous Systems

- Autonomic Scheduling Strategies without User-supplied Performance Estimations are Feasible and Efficient



progress

- **Resource Awareness**
  - Enable real-world applications
- **Scheduling**
  - Evaluate new scheduling schemes for skeletons

open issues

- 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 ?

High-Performance Computing: Gossip, Lies, & Secrets

Horacio González-Vélez
Cloud Competency Centre, NCI
E: horacio@ncirl.ie
horaciogv
22-Sep-16