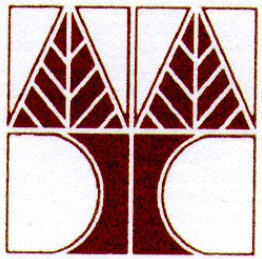
TERAFLUX

Exploiting dataflow parallelism in Teradevice Computing

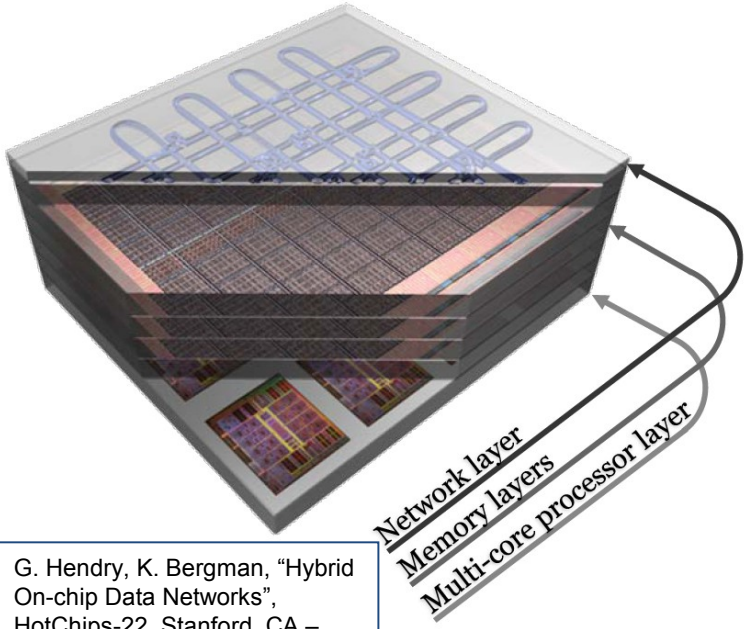
Göteborg, April 24, 2012

TERAFLUX Partners

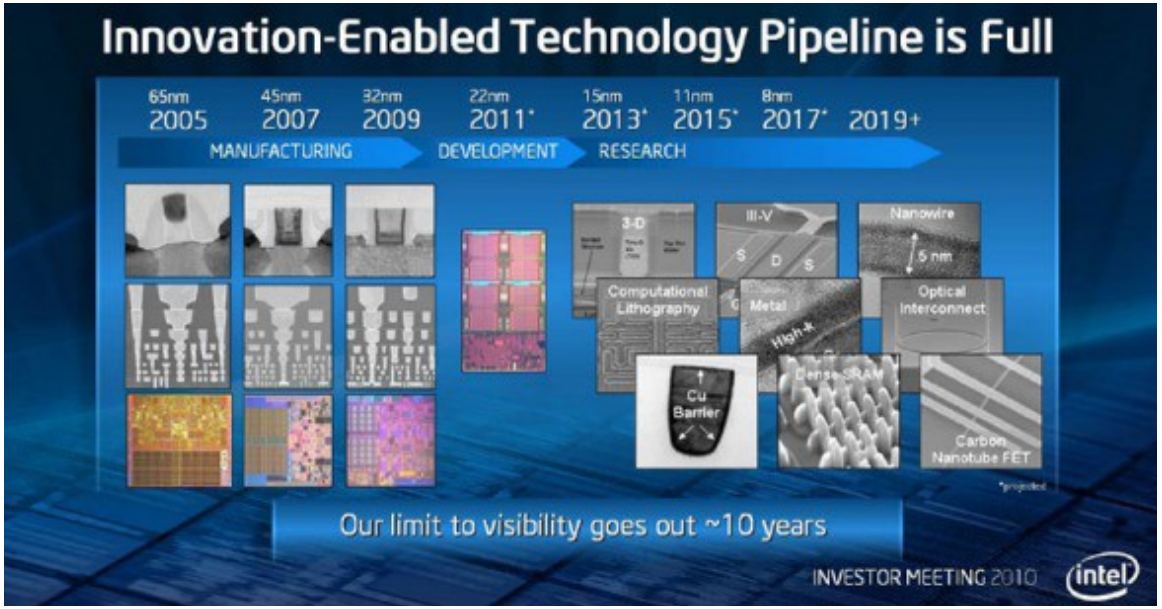
Beneficiary Number	Beneficiary Name	Beneficiary short name	Country
1 (coordina-tor)	Università degli Studi di Siena	UNISI	Italy
2	Barcelona Supercomputing Center	BSC	Spain
3	CAPS Enterprise	CAPS	France
4	Hewlett Packard	HP	Spain
5	INRIA	INRIA	France
6	Microsoft	MSFT	Israel
7	THALES	THALES	France
8	University of Augsburg	UAU	Germany
9	University of Cyprus	UCY	Cyprus
10	The University of Manchester	UNIMAN	UK

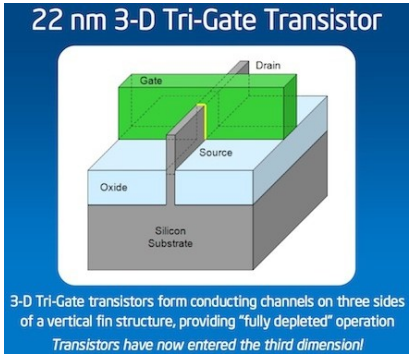



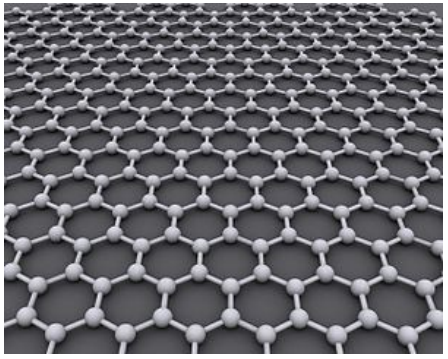
Technology Scenarios

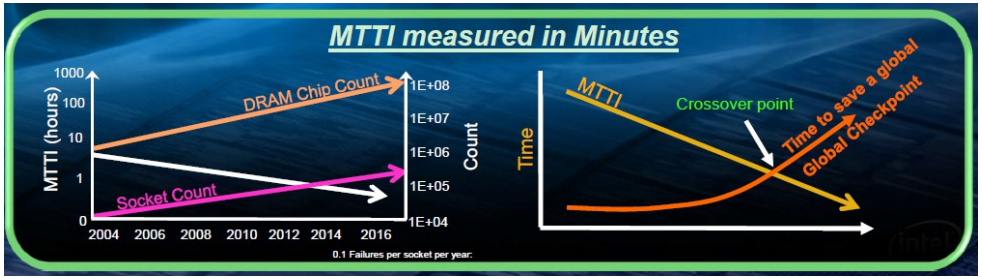


G. Hendry, K. Bergman, "Hybrid On-chip Data Networks", HotChips-22, Stanford, CA – Aug. 2010



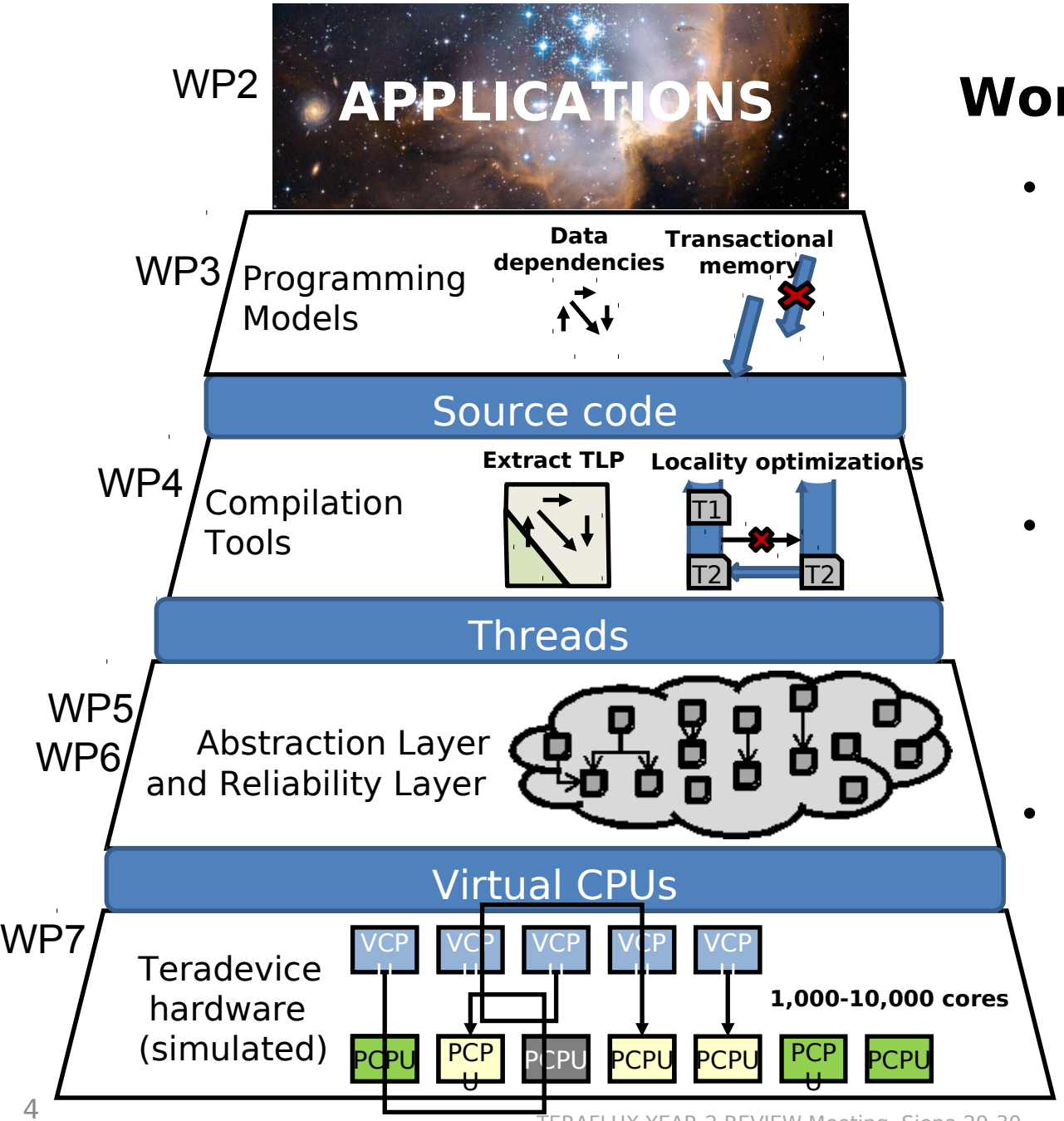






Pawloski, May 2011, Exascale Seminar, Ghent

Working Hypothesis



- Tera-transistor chips? Challenges:
  - (at least) programmability, complexity of design, reliability
- TERAFLUX context
  - High performance computing and general-purpose applications
- TERAFLUX scope
  - Exploiting dataflow principles at every level of abstraction

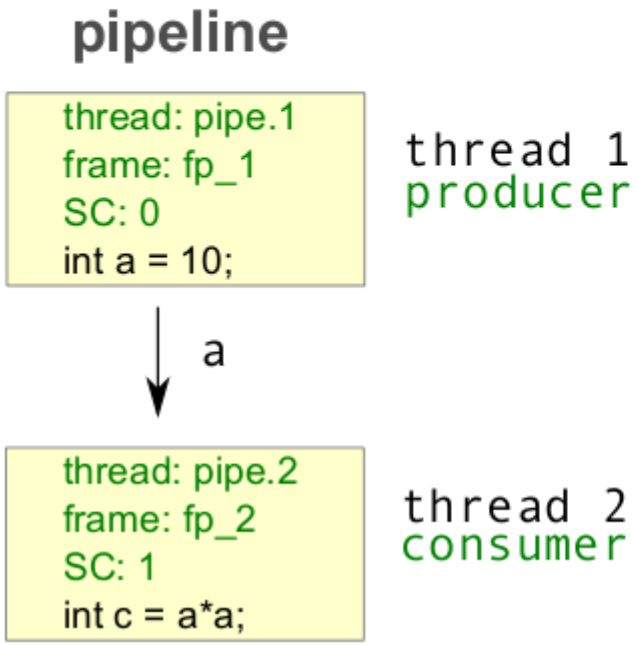
DATAFLOW

A scheme of computation in which an activity is initiated by presence of the data it needs to perform its function

Jack Dennis

Dataflow Threads

```
void pipe (){
  int a = 10;
  int c = a*b;
}
```




```
void pipe.entry(){
  fp_1 = tcreate(pipe.1, 0, sz);
  fp_2 = tcreate(pipe.2, 1, sz);
  fp_1->fp_2 = fp_2;
}
void pipe.1 (){
  cfp = tget_cfp();
  fp_2 = cfp->fp_2;
  int a = 10;
  fp_2->a = a;
  tdecrease (fp_2);
}
void pipe.2 (){
  cfp = tget_cfp();
  a = cfp->a;
  c = a*a;
}
```

Pillars

COMPATIBLE WITH EXISITING ISAs (**x86**)

MANYCORE FULL SYSTEM SIMULATOR (**COTSon**)

REAL WORLD APPLICATIONS

BEYOND DATAFLOW: **TRANSACTIONAL MEMORY**

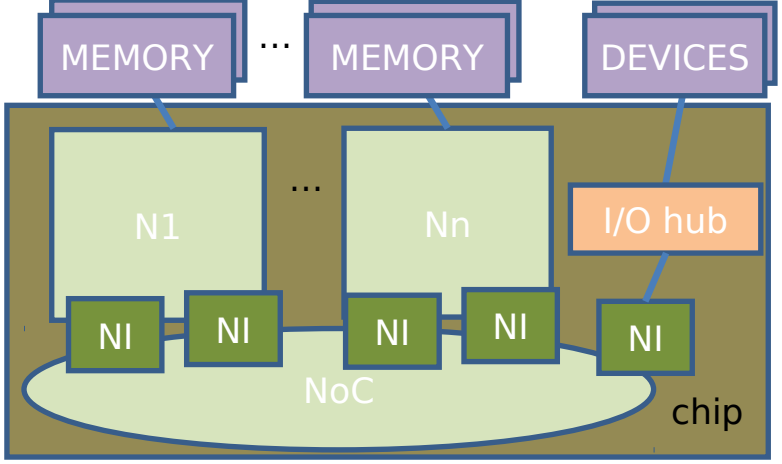
EFFICIENCY AND **PRODUCTIVITY** LANGUAGES: C+PRAGMAS, SCALA

GCC-BASED TOOL-CHAIN

OFF-THE-SHELF COMPONENTS FOR CORES, OS, NoC, MEMORY HIERARCHY

FDU AND **TSU**: Fault Detection Unit and Thread Scheduling Unit

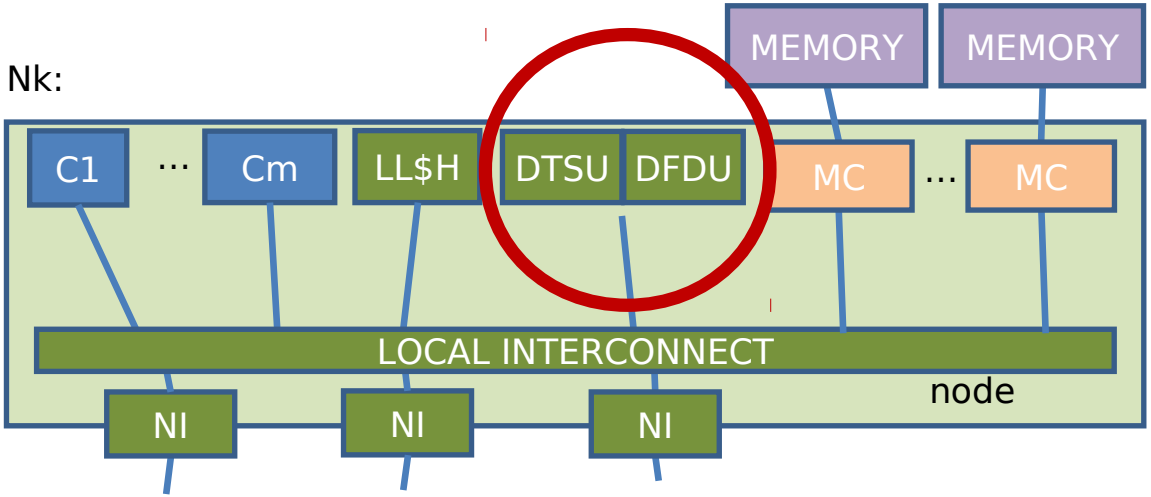
TERAFLUX Architectural template



n = # of nodes
m = # of cores per node

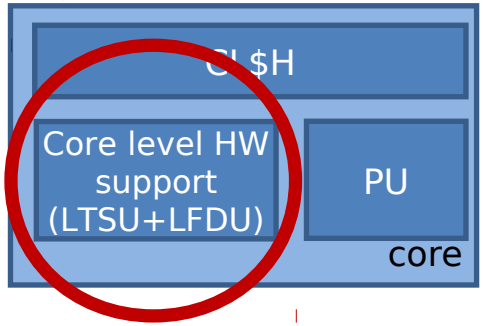
Nk = k-th Node (1..n)
NI = Network Interface
NoC = Network on Chip

Nk:



Cj = j-th core (j=1..m)
MC = Memory Controller
DTSU = Distributed TSU
DFDU = Distributed FDU
LL$H = Last Level Cache Hierarchy

Cj:



CL$H = Core Level Cache Hierarchy
PU = Processing Unit
LTSU = Local TSU
LFDU = Local FDU

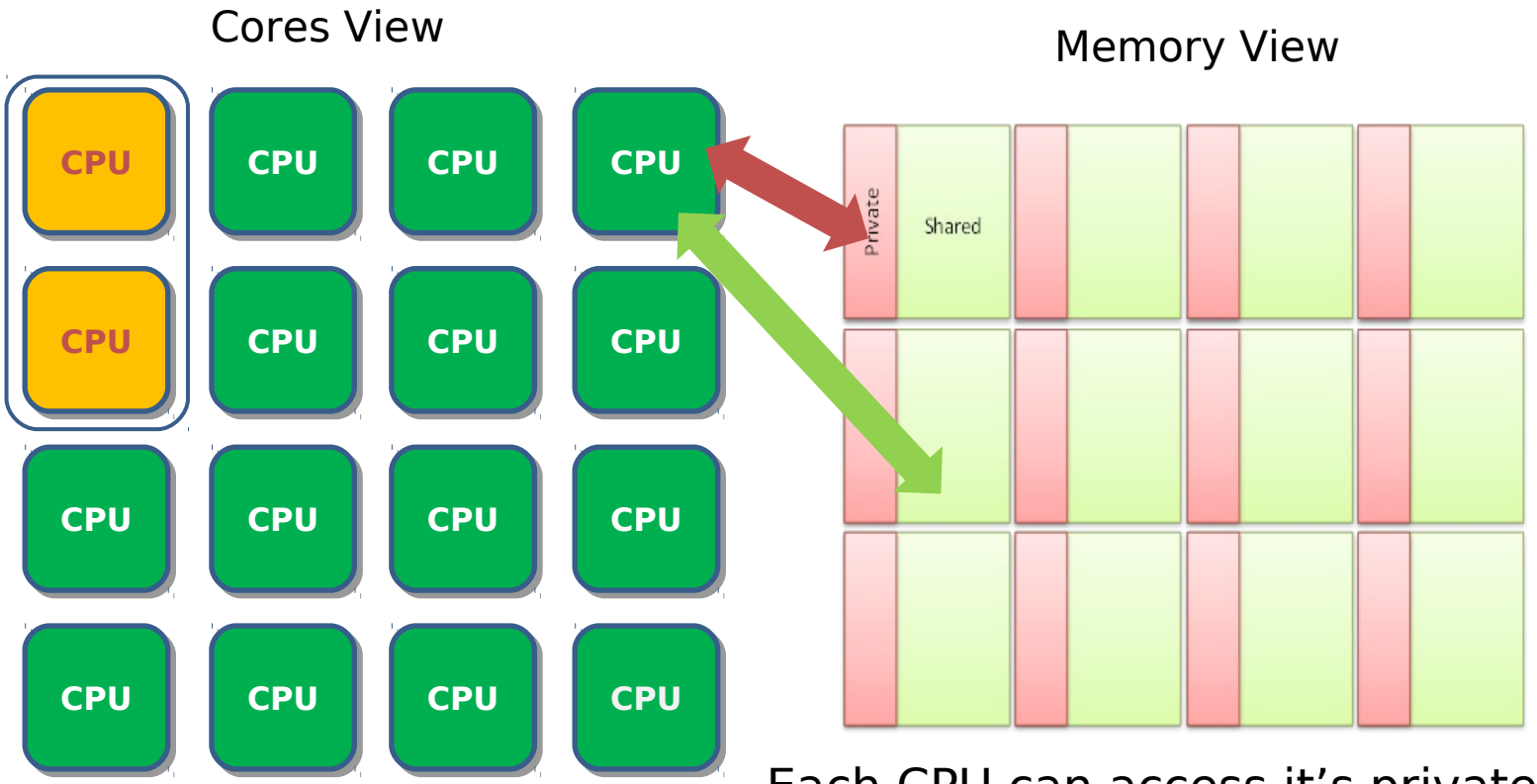
CHIP LEVEL

NODE LEVEL

CORE LEVEL

TERAFLUX

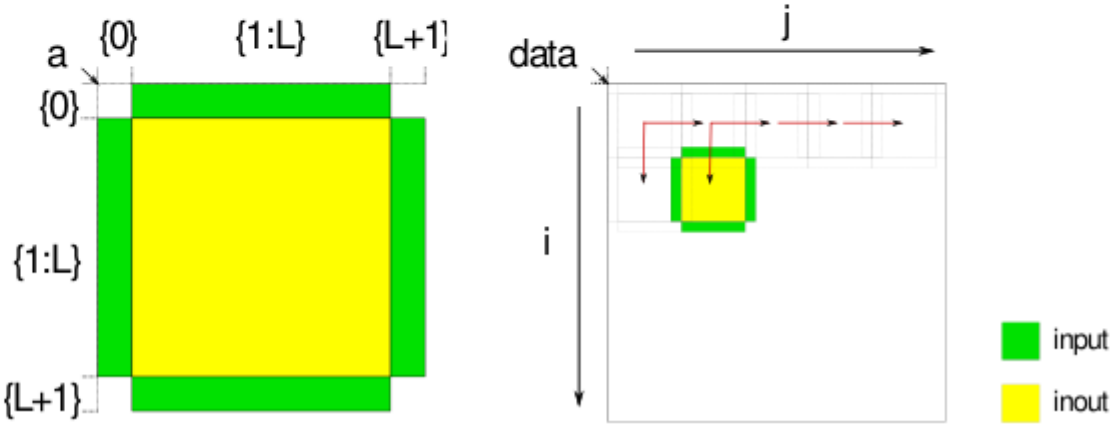
Operating System Low Level Core/Memory Map



Each CPU can access it's private memory
All shared memories can be accessed as one virtual linear address space by the DMA

Programming Models
StarSs Example: Gauss-Seidel

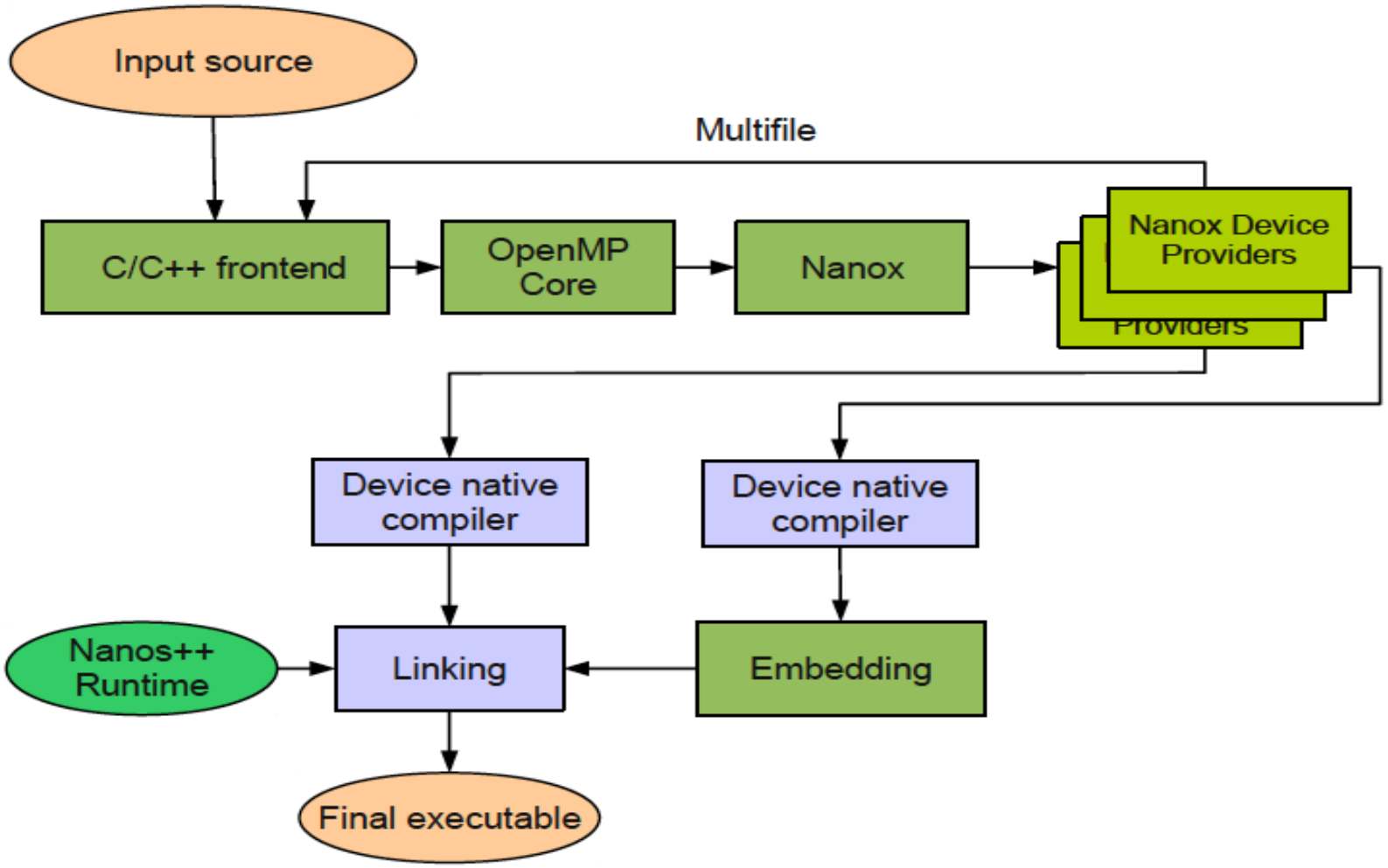
```
#pragma css task input(a{0}{1:L}, a{L+1}{1:L}, a{1:L}{0}, a{1:L}{L+1}) inout(a{1:L}{1:L})
void gauss_seidel (double a[N][N]) {
  for (int i=1; i<=L; i++)
    for (int j=1; j<=L; j++)
      a[i][j] = 0.2 * (a[i][j] + a[i-1][j] + a[i+1][j] + a[i][j-1] + a[i][j+1]);
}

for (int it=0; it < NITERS; it++)
  for (int i=0; i<N-2; i+=L)
    for (int j=0; j<N-2; j+=L)
      gauss_seidel(&data[i][j]);
```



[1] Josep M. Perez, Rosa M. Badia, and Jesus Labarta. 2010. *Handling task dependencies under strided and aliased references.* In Proceedings of the 24th ACM International Conference on Supercomputing (ICS '10). ACM, New York, NY, USA, 263-274.

Programming Models

OMPSs framework, StarSs language



Programming Models
Evolution of HMPP

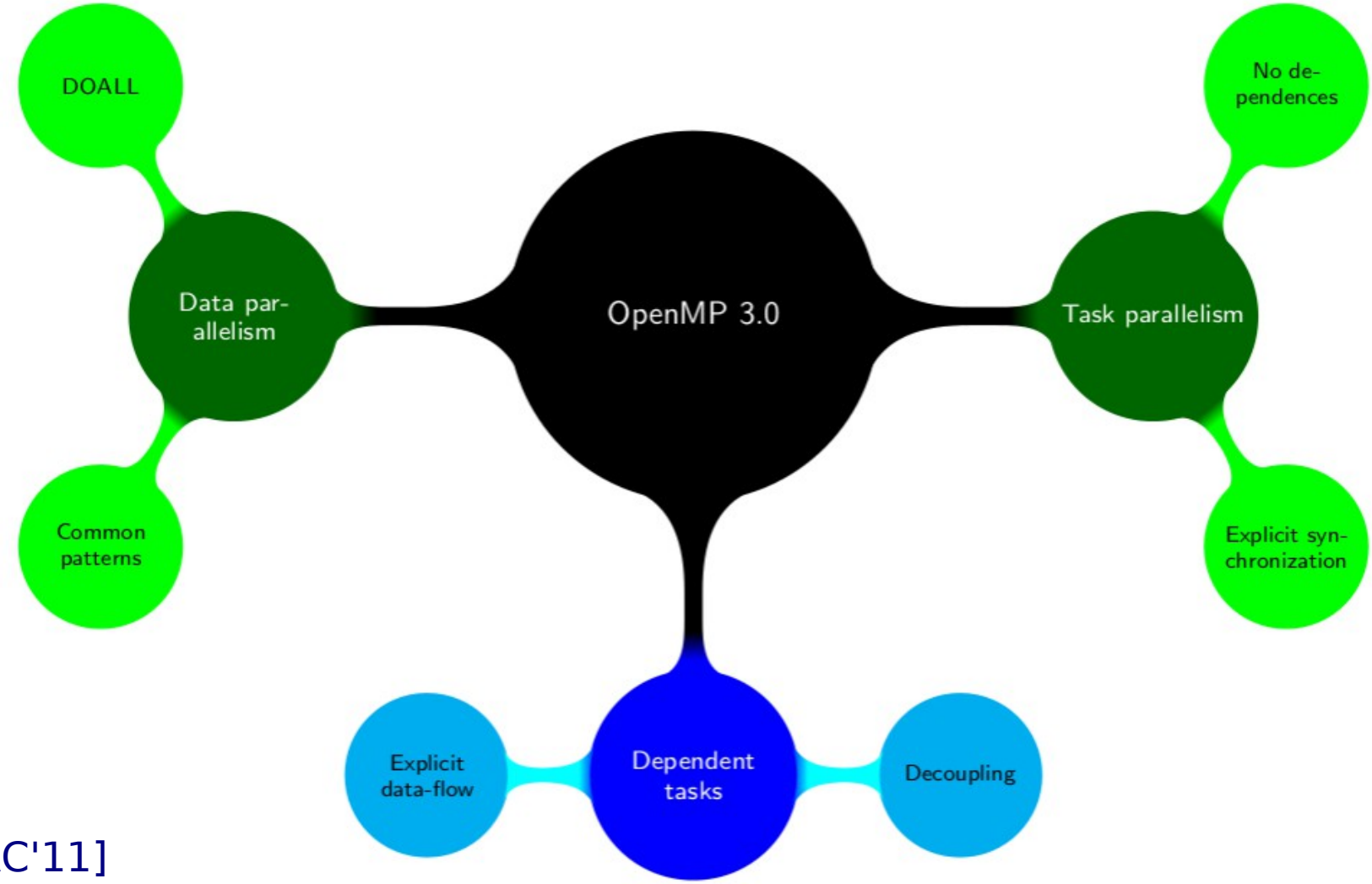
- CAPS HMPP Workbench for TERAFLUX
  - TERAFLUX as a manycore accelerator device
  - Automatic offloading, management of the memory hierarchy
  - Loop transformations for codelets (task-level)
- Evolution of HMPP

  Direct codelet-to-codelet communication

  Collection-based parallelism with a map operator on codelets

  OpenHMPP consortium

Programming Models
Streaming OpenMP



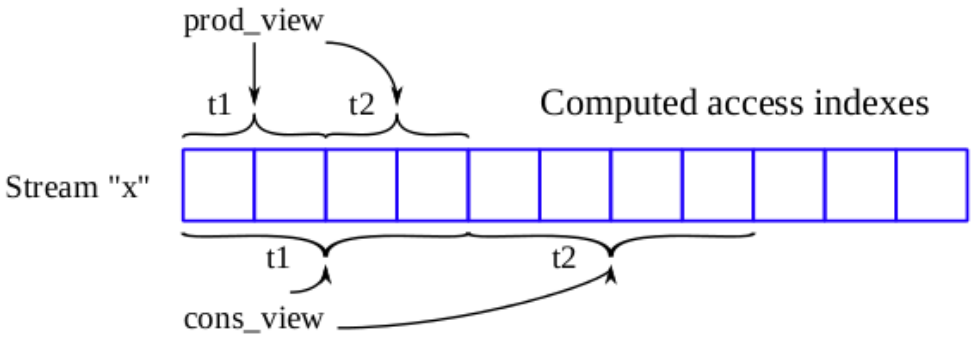
[HiPEAC'11]

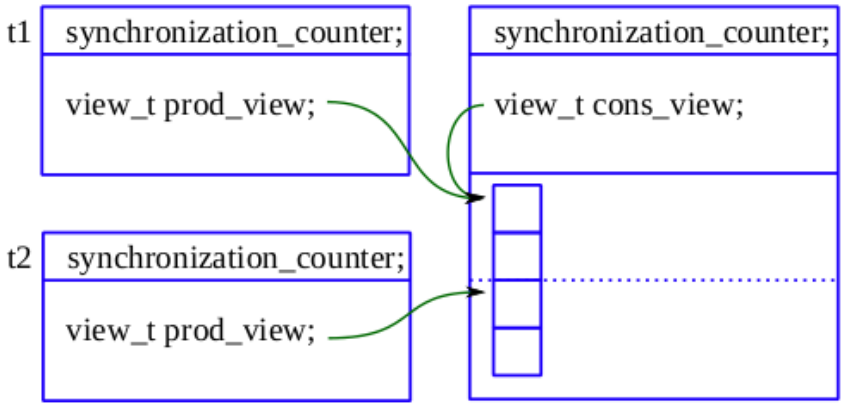
Programming Models

Streaming OpenMP to dataflow execution model

```
int prod_burst = ..., cons_burst = ...;
int x, prod_view[prod_burst], cons_view[cons_burst];

#pragma omp task output (x >> prod_view[prod_burst])
    prod_view[0..prod_burst-1] = ...;

#pragma omp task input (x << cons_view[cons_burst])
    ... = cons_view[0..cons_burst-1];
```

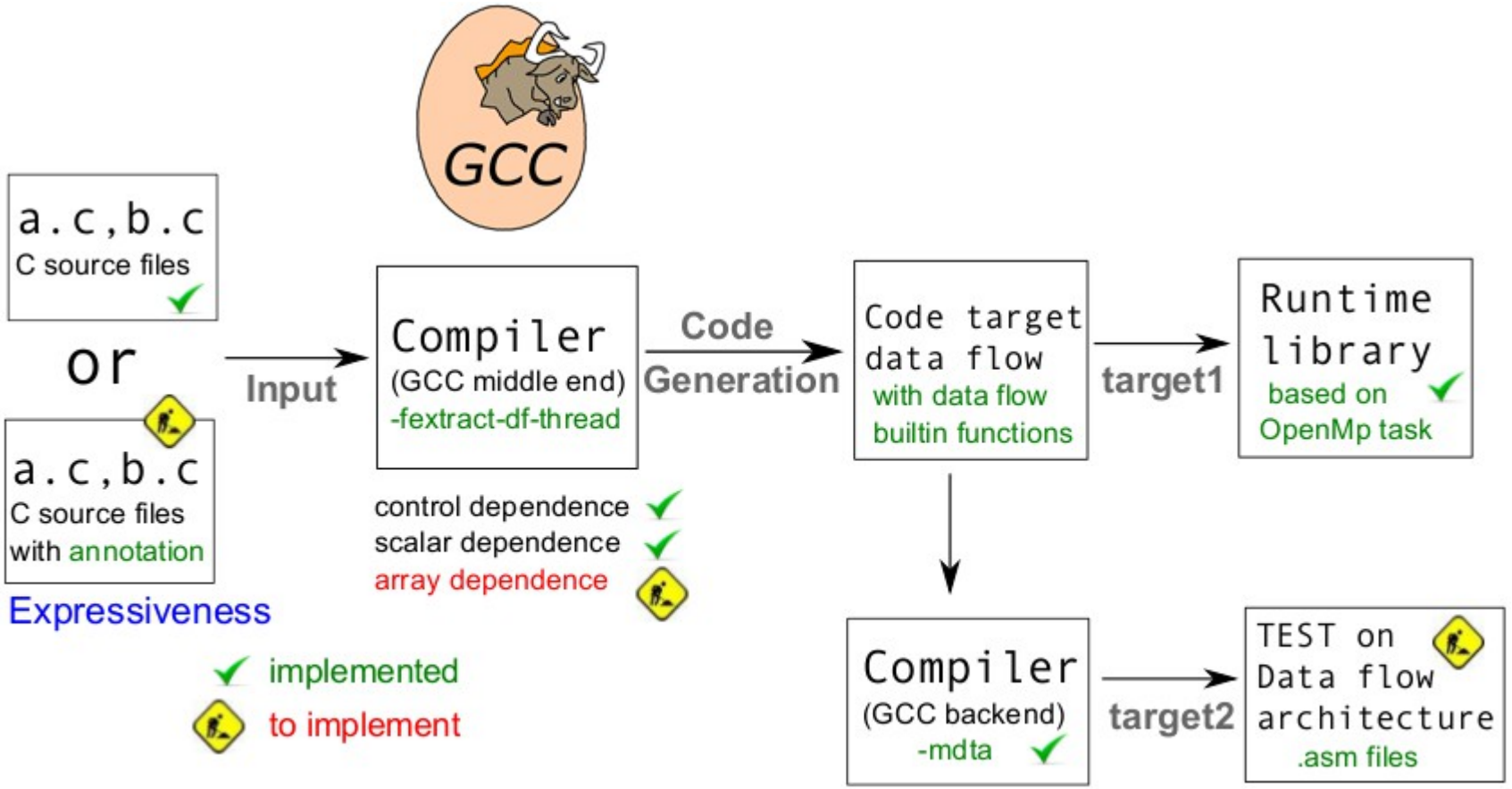
↓ Dynamically resolve flow dependences between task activations ↓



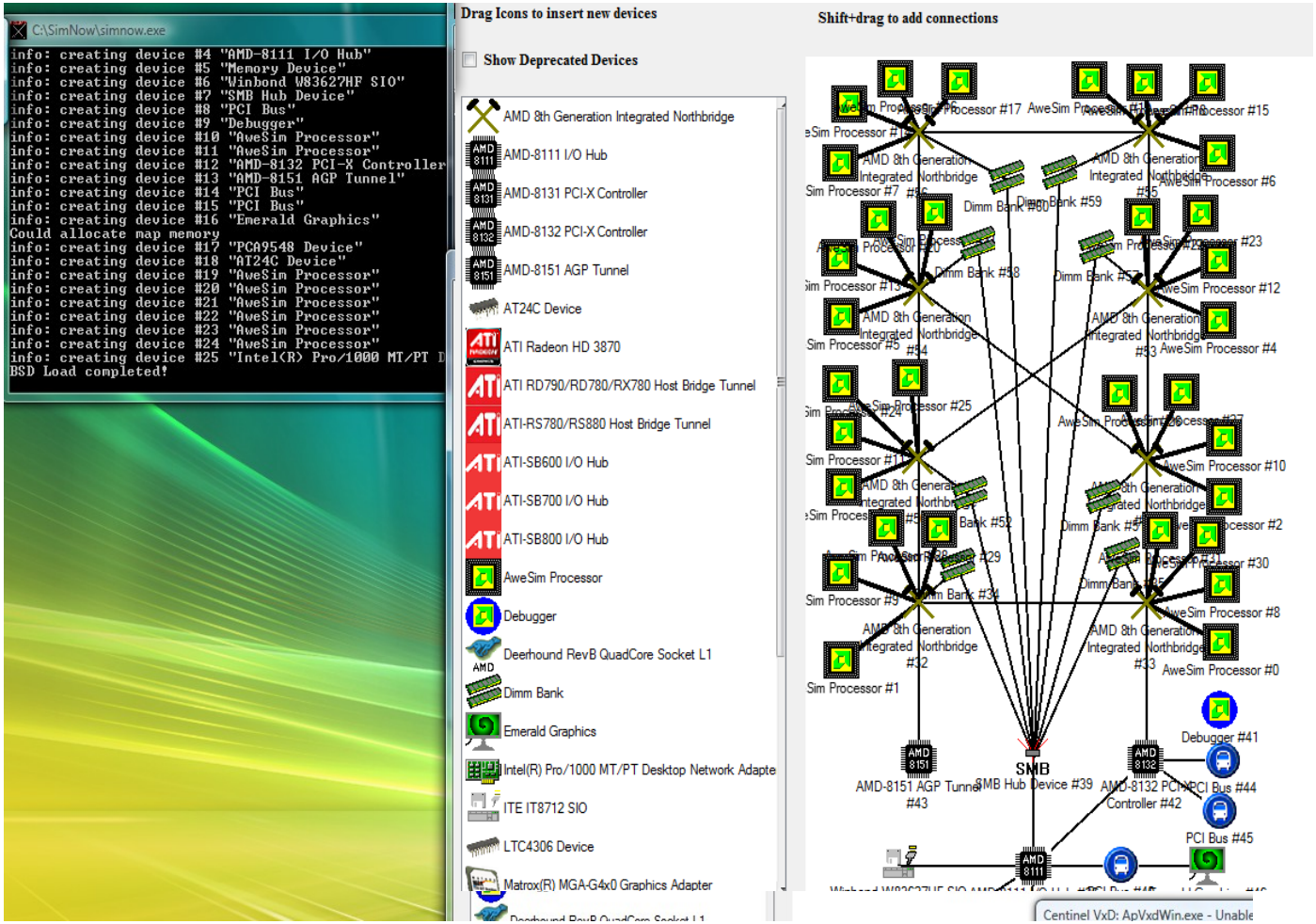


Compilation for Dataflow Threads

Automatic DF Thread Extraction



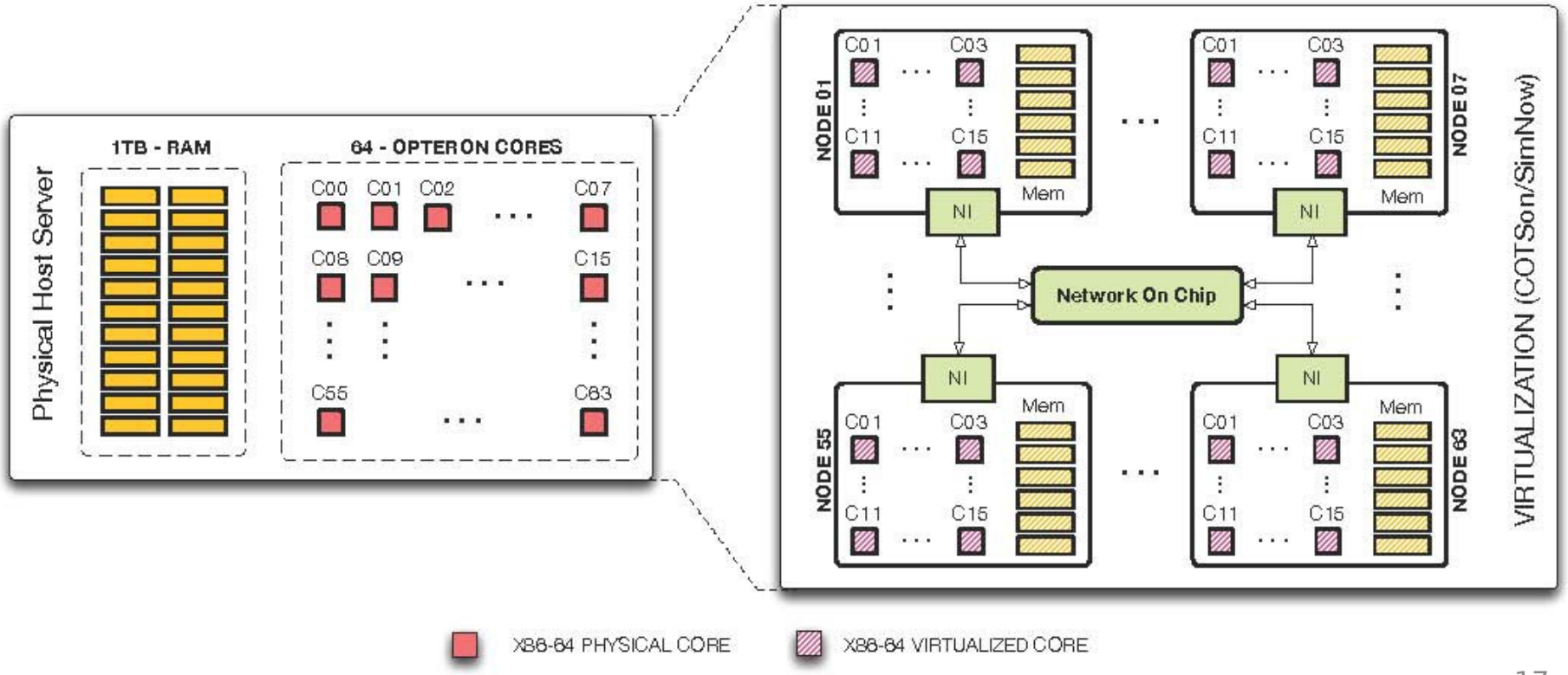
AMD SIMnow and COTSon



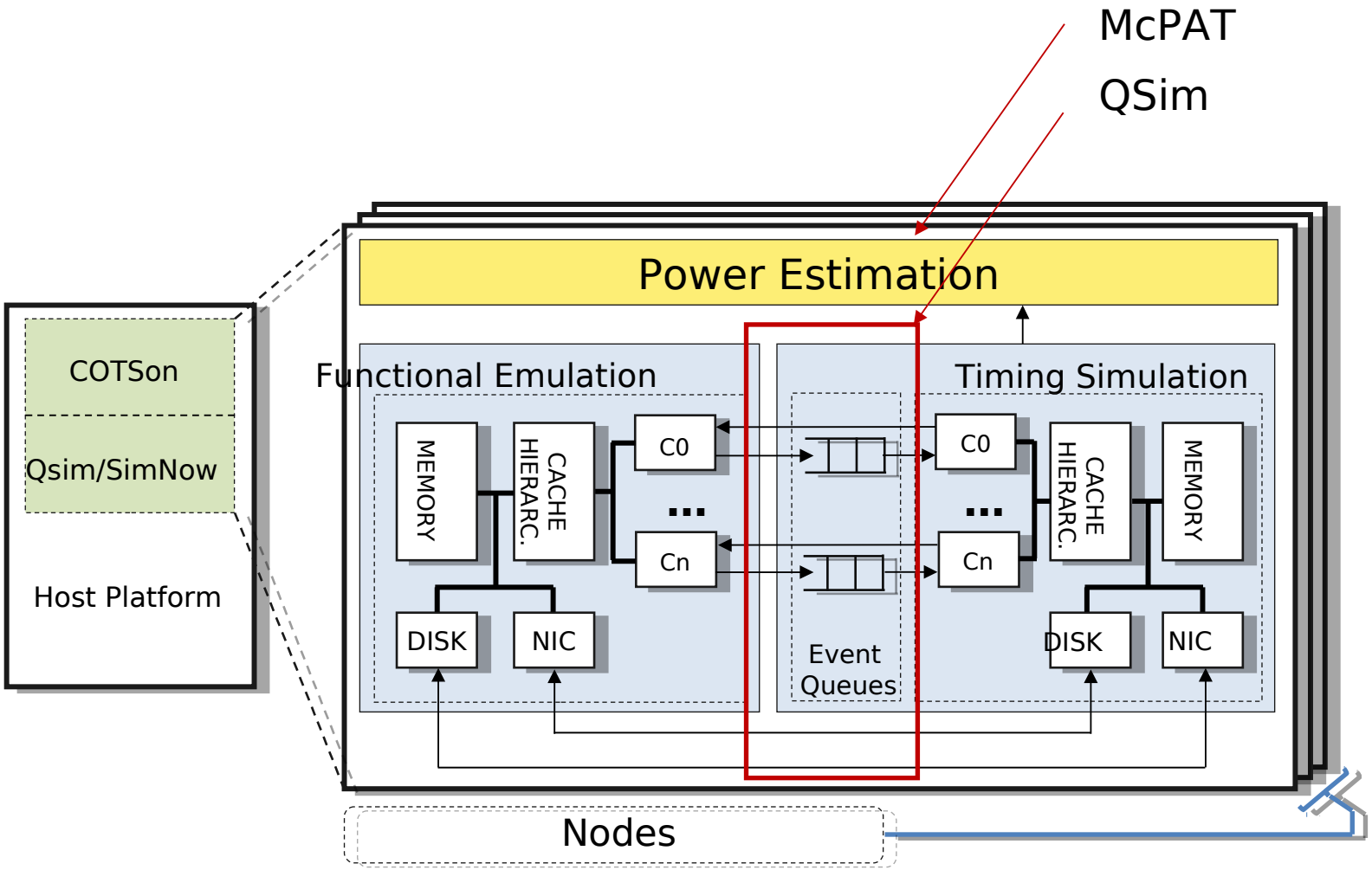
Example: 1024 cores setup

DL-Proliant DL585 G7
AMD Opteron 6200
4 sockets, 64 cores total
1TB DRAM

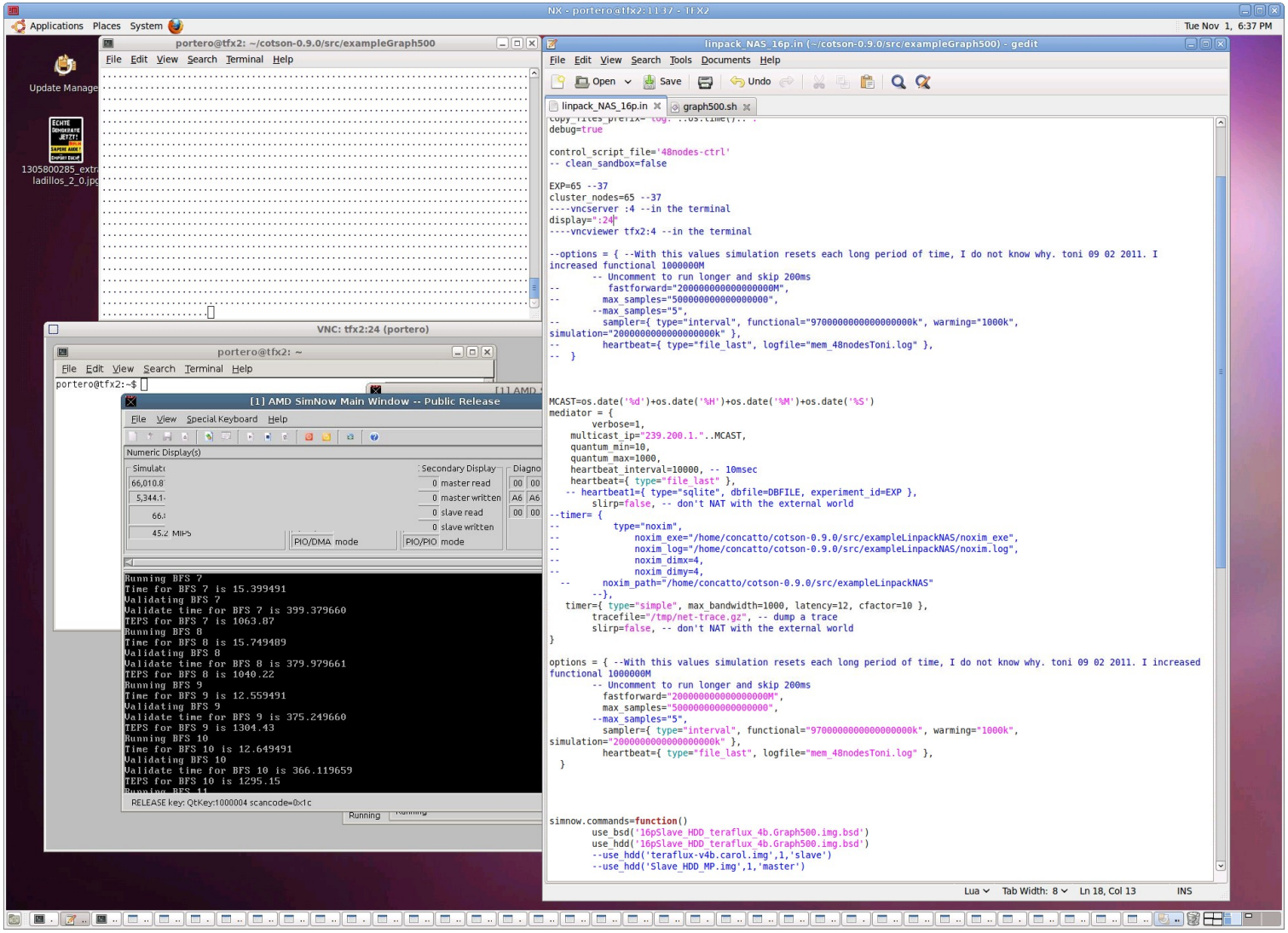
TERAFLUX System instance
AMD Opteron-L1_JH-F0 (800Mhz)
64 nodes, 16 cores each, 1024 cores total
256M DRAM per core



COTSon+QSim+McPAT



Running Graph500 on 1024 cores (64x16)



TERAFLUX Impact

- “To increase and accelerate the impact of FET research projects by cooperating with non-EU partners of excellent global standing. It targets the extension of ongoing FET projects with complementary research activities in which collaboration with non-EU research partners brings significant added value”

- **EC has approved a TERAFLUX extension for an additional partner (U. of Delaware – Prof. Guang Gao) – 420,000 EC funding**
  - **Period from April 1st 2012 to December 31st 2013**
  - **Aligned with TERAFLUX “timetable”**

FUTURE AND EMERGING TECHNOLOGIES
PROJECT N. 249013

SEVENTH FRAMEWORK PROGRAMME
FET proactive 1 (ICT-2009.8.1)
Concurrent Tera-Device Computing

TERA^{F}LUX

Exploiting dataflow parallelism in Teradevice Computing

http://teraflux.eu