What to expect from the people who want to move forwards

Sverre Jarp

8 July 2011

CERN openlab

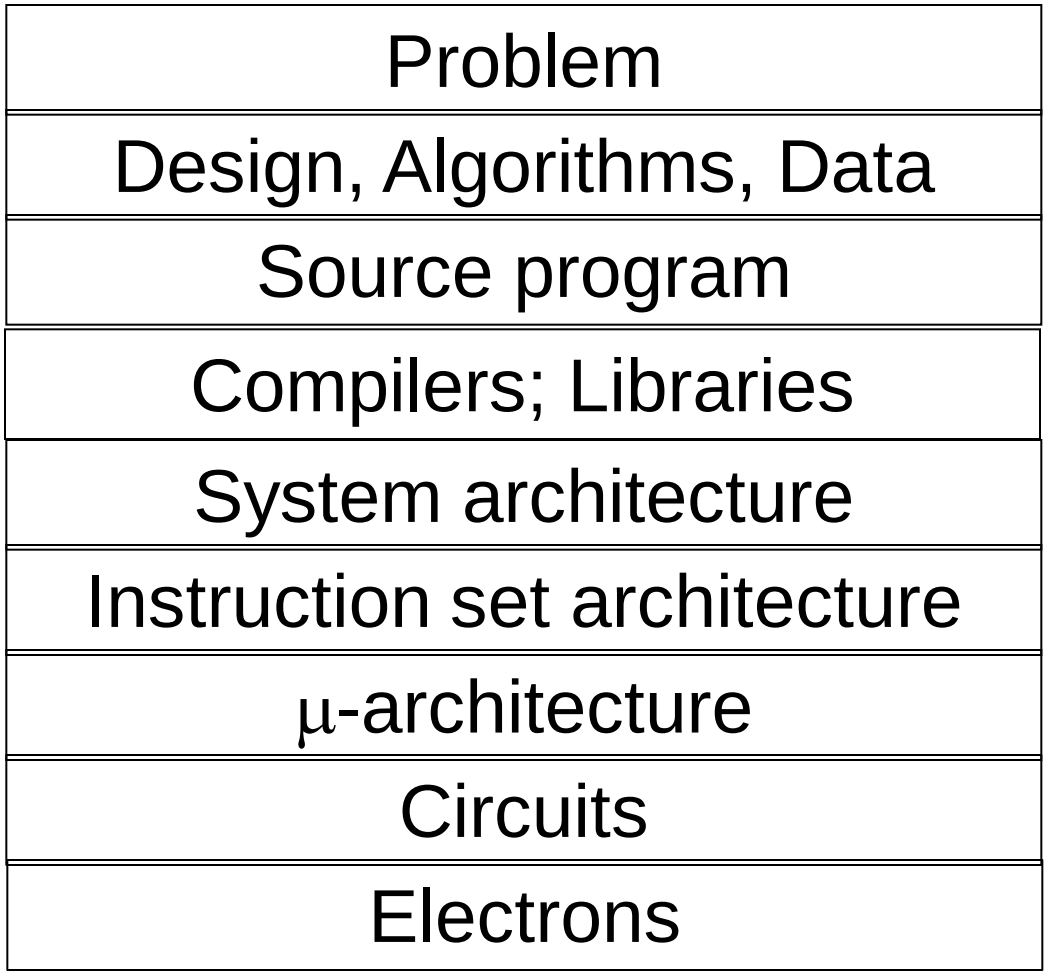
Second International Workshop for future challenges in tracking and triggering concepts

CERN openlab

How best to move forward?

The complex story of computing

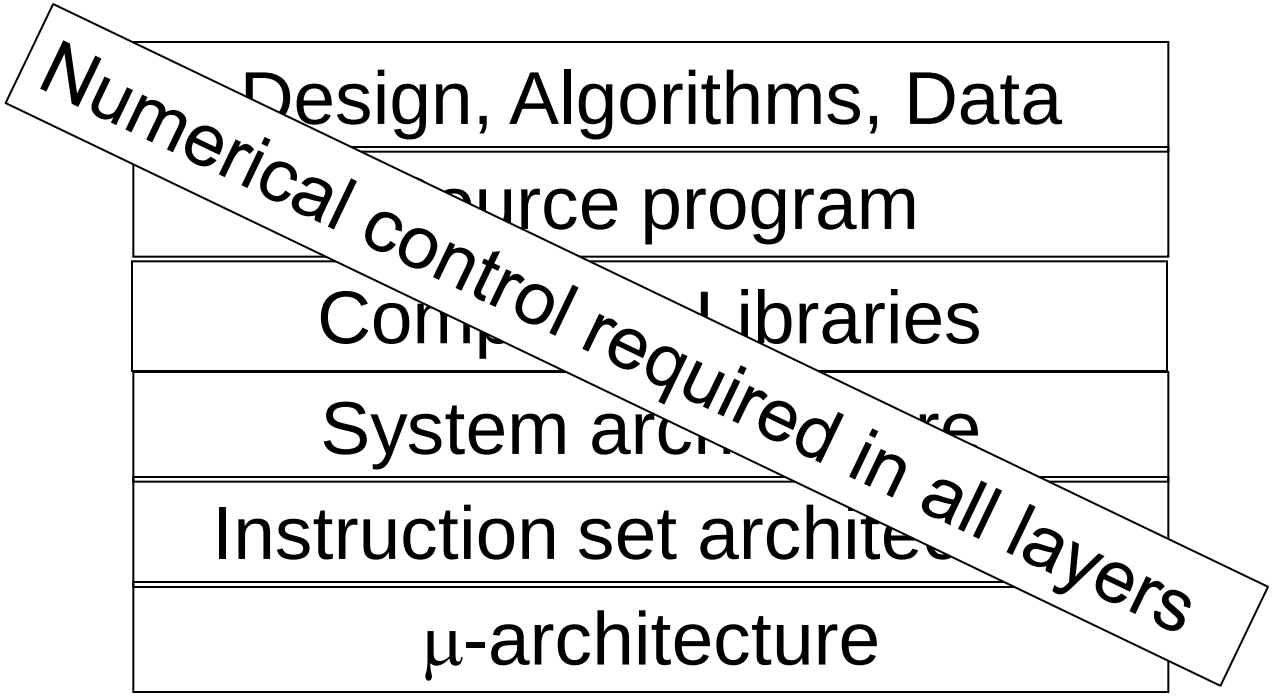
- **In multiple layers:**



Adapted from Y.Patt, U-Austin

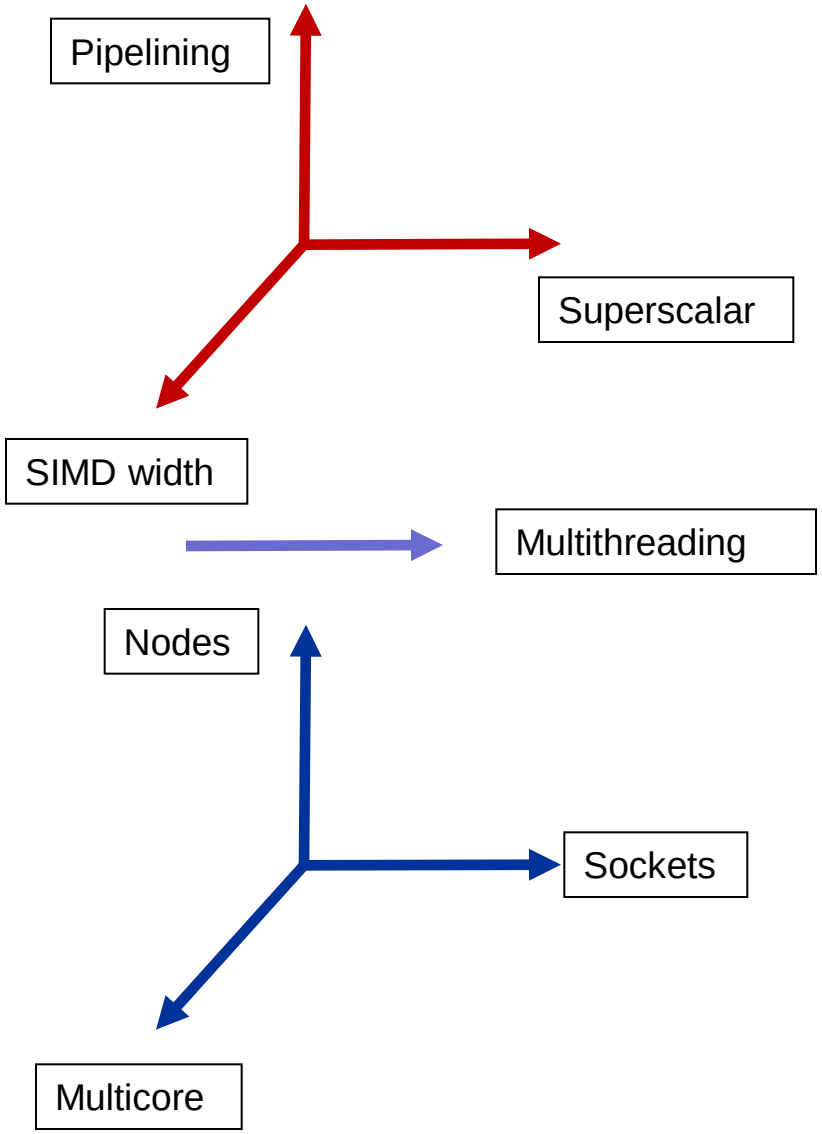
Take numerical accuracy (as an example)

- **Affects most of the layers:**
  - Requiring 10^{-6} may be fine
    - 10^{-18} (i.e. bit for bit reproducibility, may be impossible)
  - In the Exascale era:
    - You may not even get the same numerical result from one run to the next (on the same hardware)



Seven dimensions of performance

- **First three dimensions:**
  - Pipelining
  - Superscalar
  - Computational width/SIMD
- **Next dimension is a "pseudo" dimension:**
  - Hardware multithreading
- **Last three dimensions:**
  - Multiple cores
  - Multiple sockets
  - Multiple compute nodes



Seven multiplicative dimensions

- **First three dimensions:**
  - Pipelining
  - Superscalar
  - Computational width/SIMD
- **Next dimension is a "pseudo" dimension:**
  - Hardware multithreading
- **Last three dimensions:**
  - Multiple cores
  - Multiple sockets
  - Multiple compute nodes

- **Need to understand overall hardware potential**
- **Where are we on the scale ?**
  - 10% ?
  - 90% ?
- **Have we made the same effort on the CPU and the GPU?**

When the HW potential is fully known

- **Example from matrix multiply:**
  - "Optimizing matrix multiplication for a short-vector SIMD architecture – CELL processor"
    - J.Kurzak, W.Alvaro, J.Dongarra
    - Parallel Computing 35 (2009) 138–150

In this paper, single-precision matrix multiplication kernels are presented implementing the $C = C - A \times B^T$ operation and the $C = C - A \times B$ operation for matrices of size 64x64 elements. For the latter case, the performance of 25.55 Gflop/s is reported, or 99.80% of the peak, using as little as 5.9 kB of storage for code and auxiliary data structures.

CERN openlab

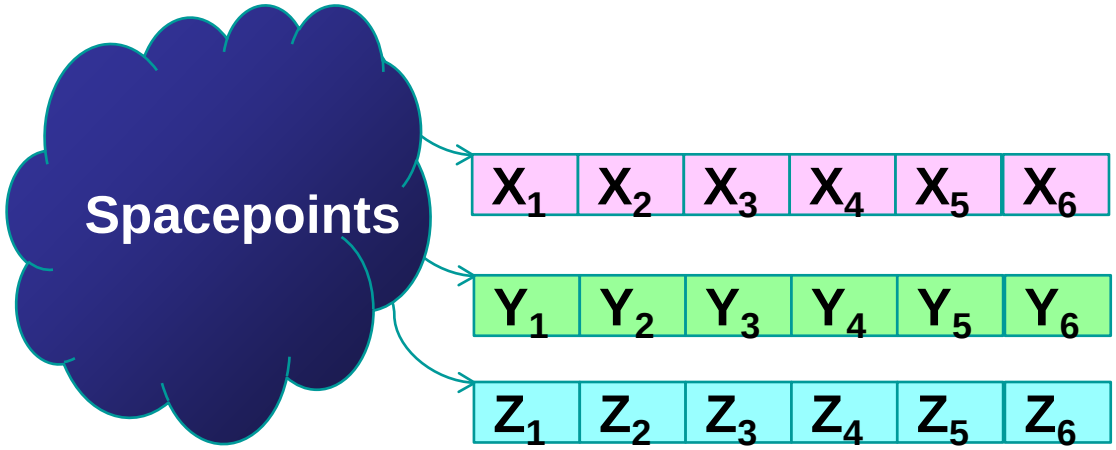
Vector computing

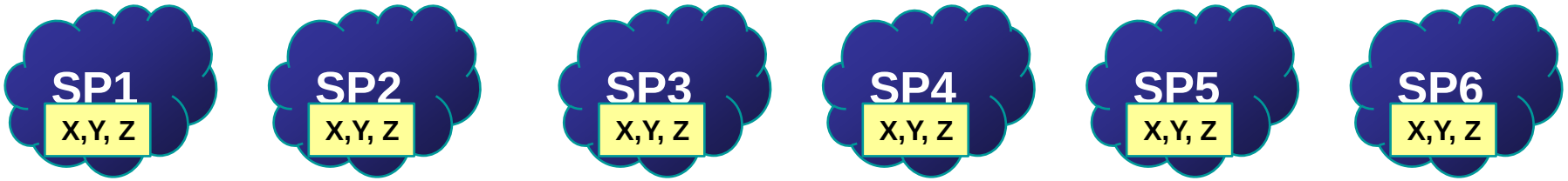
Every computer is now a vector computer

- **DP [SP] SIMD sizes:**
  - 2x [4x] last ten years → 4x [8x] now (2011)
  - 8x [16x] in MIC architecture
    - This is too much performance to leave on the table
    - Even if we do not achieve (anywhere near) 100% of peak
- **HEP and vectors: Too little common ground**
  - Practically all attempts in the past failed.
    - w/CRAY, CYBER 205, IBM 3090-Vector Facility, etc.
    - Interesting reading: Dekeyser J 1987 “Vectorization of the GEANT3 geometrical routines for a Cyber 205”
- **But we need to try once more!**
  - 25 years later!

SoA versus AoS

- In general, compilers and hardware prefer the former!
- Structure of Arrays (SoA):
- Array of Structures (AoS):





CERN openlab

Which compilers to adapt?

Compilers on the market (1)

- **Linux/open source:**
  - **GNU** compiler suite. Fortan and C/C++ [4.6.1]
    - http://gcc.gnu.org/
  - **LLVM** (C/C++) compiler framework [2.9]
    - Originated from U. of Illinois
    - Now supported by Apple
      - http://www.llvm.org/
  - **Open64** compiler suite. C/C++/Fortran [4.2.4]
    - Derived from the SGI MIPS, IA-64 compiler
      - http://www.open64.net/
    - Now also supported by AMD [4.2.5]
      - http://www.amd.com/open64

Compilers on the market (2)

- **Commercial/Linux or Windows:**
  - **Intel compiler suite** (C/C++/Fortran for IA-32, Intel64, and IA-64); http://www.intel.com/ [12.0, 12.1 in beta]
  - ST Microelectronics/**Portland Group (PGI)** (C/C++/Fortran compilers; http://www.pgroup.com/ [11.5]
  - **Pathscale compilers** (Now owned by NetSyncro.com). Also derived from SGI's compilers. (C/C++/Fortran); http://www.pathscale.com/
  - **Microsoft** C/C++ compiler; http://www.microsoft.com/
  - **Lahey/Fujitsu** Fortran compiler; http://www.lahey.com/
  - **NAG Fortran** Compiler; http://www.nag.com/

List of optimisation flags

- **Common Linux compilers:**

	PGI	Intel	GNU	Open64
Global optimisations	-O2	-O2	-O2	-O2
Aggressive opt.	-O3	-O3	-O3	-O3
Maximise performance	-fast	-fast	-Ofast	-Ofast
Interprocedural optimisation	-Mipa=fast	-ipo	-flto	-ipa
Profile-guided optimisation	-Mpfi -Mpfo	-prof-gen -prof-use	-fprofile-generate -fprofile-use	-fb-create -fb-opt
Unrolling	-Munroll	-unroll -unroll[N] -unroll-aggressive	-unroll-loops -unroll-all-loops	WOPT:unroll=N

Adapted from AMD brochure

List of vectorisation/parallelisation flags

- **Common Linux compilers:**

	PGI	Intel	GNU	Open64
Intrinsics	Yes	Yes	Yes	Yes
Autovectorisation	-Mvect[=sse]	-simd -vec (default)	-ftree-vectorize	LNO:simd=N
Report on autovectorisation	???	-vec-report[N]	-ftree-vectorizer-verbose=[N]	LNO:simd_verbose
Vector Math Library	-lacml	-lsvml	mveclibabi=acml/svml	-lacml
OpenMP	-mp	-openmp	-fopenmp	-mp -openmp
Autoparallelisation	-Mconcur	-parallel		-apo

Adapted from AMD brochure

List of flags (for accuracy control)

- **Common Linux compilers:**

	PGI	Intel	GNU	Open64
Accuracy control	-Kieee	-fp-model	-ieee-fp	-fp-accuracy
Fast transcendentals		-fast-transcendentals		
Relaxed FP control	-Mfprelaxed		-ffast-math	

Adapted from AMD brochure

ACML versus SVML (Vector routines)

- **Similar, but not identical**
  - Note that div and sqrt are native instructions

	ACML (SSE only)	SVML (also AVX)
Group 1		invsqrt cbrt, invcbrt
Group 2	log, log10, log2 exp pow (SP only)	log, log10, log2 exp, exp2 pow
Group 3	sin, cos, sincos	sin, cos, sincos, tan asin, acos, atan, atan2
Group 4		sinh, cosh, tanh

The big issue with compilers

- **Which one(s) to choose**
  - Intel is clearly storming ahead with advanced support for vector/parallel hardware
  - As shown, the others also have support, but not at the same level:
    - CilkPlus/Advanced Vector Syntax, SIMD pragmas, Advanced vector math library, etc.
- **Our dilemma:**
  - Choose the lowest common denominator?
  - Adopt new compiler features as quickly as possible?

Questions at the end

- **How to improve (in our community):**
  - Control of numerical results
    - Whether we are using vector/parallel hardware, CPUs/GPUs, 53/64-bit mantissa, different compilers/libraries, etc.
  - Exploitation of the vector capabilities in hardware
    - "Every computer is a vector computer"
  - Use of array syntax / Structures of Arrays
  - Continued good use of parallelism with memory under control
    - It is rather incomprehensible that we need 1-10 GB of memory to computer an event that is 1 MB in size
  - A related question:
    - How to move all the good prototypes back into mainstream source?

CERN openlab

BACKUP

Q & A

CERN
openlab