

Computational Instrumentation for Nanoscience

T. C. Schulthess¹, P. T. Cummings², and G. M. Stocks³

¹Computer Science and Mathematics Division; ²Center for Nanophased Materials Sciences; ³Metals and Ceramics Division; Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831-6164

High Performance Computing (HPC) of the 1990s

- HPC facilities based on commodity hardware
- Users develop/port/run codes with minimal support



User Community
mostly individual
researchers, small groups
(exceptions: climate,
chemistry)

- Efficiency of HPC in science is very low
 - Many codes run a low percentage of peak
 - Researchers spend most of their time developing / porting / optimizing codes

Recent Trends in HPC: "Extreme Computing"

- Deliver 100-1000 times more performance with specialized hardware not commonly available



~10⁴ vector proc.



~10⁵ super scalar proc.

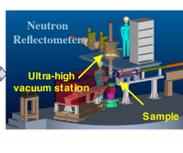
- Capability Computing (few use entire resource)
Applications *must* scale and run with high efficiency

Capability Computing for Science

- Consider analogy with scientific user facilities



Facility



Instrumentation

User
Community



Facility



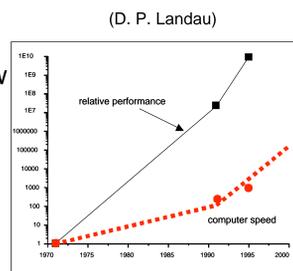
Instrumentation

User
Community

- There is a need to develop the domain specific scientific *instrumentation* for facilities that deliver capability computing.

Nanoscience and the National Leadership Computing Facility (NLCF): opportunities and things to keep in mind

- Nanoscience will likely deliver high impact science and technology.
 - Computational nanoscience did influence magnetic data storage technology, a prime example where nanoscience is already having major impact in technology.
- Nanoscience has a proven record of delivering high efficiency computation.
 - Materials / condensed matter sciences first to reach sustained Gflop/s and Tflop/s.
 - First efficient applications on Cray X1: QMC/DCA for 2D Hubbard, Molecular Dynamics for nano-tribology
- Goal is science and computers are tools
- Highest payoff from new algorithms/methods
 - Hardware improvements are necessary but *not sufficient* for leadership computing!
- Most algorithmic / methodological improvements happen in "small shops"



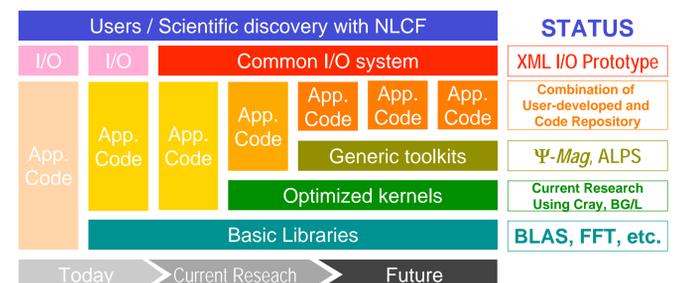
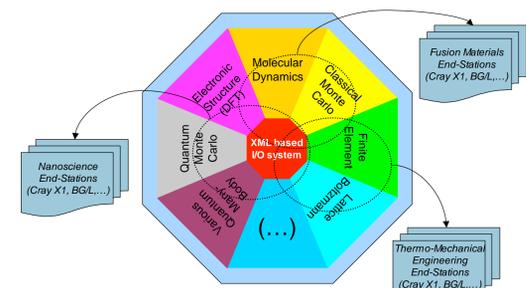
- Computer lifetime is 3-5 Years but it takes decades to develop methods and codes
 - NLCF hardware is a moving target!
- Nanoscience continues to be an emerging field covering many domains in materials, condensed matter, and biological sciences
 - Computational challenges are moving targets!

Engage the best and brightest! Integrate methods and algorithm development with leadership computing

Ergo, a robust software architecture optimized for rapid evolution, as well as performance, is essential to achieve NLCF objectives in nanoscience.

Building Computational Instrumentation for Nanoscience

- Build software repository around common I/O system
 - emphasis is on user
- with community and from existing codes
- Port, analyze, optimize codes for Cray X1, Cray Red Storm, IBM BG/L
- Identify and extract common kernels
- Integrate these kernels with object oriented and generic toolkits
- Rapid prototyping / development of new applications
 - Based on efficient tools
 - Tailored to NLCF hardware
- Provide expertise for those who
 - run existing applications
 - develop new models, methods, and algorithms.
- Run workshops, research laboratories jointly with CNMS



XML Input Example

```

% Crystal structure
Name = BCC
Vectors =
0.5 0.5 0.5
0.5 -0.5 0.5
0.5 0.5 -0.5
Sites = 1
Fe 0.0 0.0 0.0
Symmetry points = 4
G 0.0 0.0 0.0
H 0.5 0.5 0.0
I 0.0 1.0 0.0
F 0.5 0.5 0.5
% Elements (id, name, tci file)
Elements = 2
24 Cr Cr.tci
26 Fe Fe.tci
% Bulk band structure (BBS)
BBS symmetry points = 4
G H I F
BBS point density = 20
% Surface Brillouin zone integration
Irreducible Brillouin zone = T
Brillouin zone point density = 25
    
```

```

<!-- program specific -->
<!-- crystal registration, use symmetry = true -->
<!-- TBT -->
<!-- Band structure -->
<!-- sym_points = 0.0 0.0 0.0 -->
<!-- sym_points = 0.5 0.5 0.5 -->
<!-- sym_points = 1.0 0.0 0.0 -->
<!-- sym_points = 0.5 0.5 0.5 -->
<!-- TBT -->
<!-- program specific -->
<!-- crystal -->
<!-- id = Atom001 -->
<!-- name = Fe -->
<!-- number = 26 -->
<!-- description ref = "Tc001" -->
<!-- id = Atom002 -->
<!-- name = Cr -->
<!-- number = 24 -->
<!-- description ref = "Tc001" -->
<!-- crystal -->
<!-- periodic3D -->
<!-- name = "BCC" -->
<!-- bravaisVector = 0.5 0.5 0.5 -->
<!-- bravaisVector = 0.5 -0.5 0.5 -->
<!-- bravaisVector = 0.5 0.5 -0.5 -->
<!-- bravaisVector = 0.5 0.5 0.5 -->
<!-- siteList -->
<!-- site -->
<!-- position = 0.0 0.0 0.0 -->
<!-- occupant ref = "Atom001" -->
<!-- site -->
<!-- position = 0.5 0.5 0.5 -->
<!-- occupant ref = "Atom002" -->
<!-- site -->
<!-- position = 0.0 0.0 0.0 -->
<!-- occupant ref = "Atom001" -->
<!-- site -->
<!-- position = 0.5 0.5 0.5 -->
<!-- occupant ref = "Atom002" -->
<!-- geometry -->
    
```

XML (eXtensible Markup Language) in a Nutshell

XML consists of:

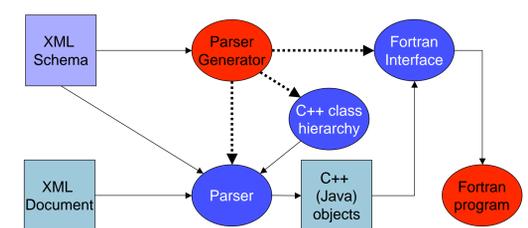
- (nested) Fields: `<tag_name> </tag_name>`
- Attributes: `<periodic3D name="BCC"></periodic3D>`
- Comments: `<!-- this is a comment -->`
- Data: `<bravaisVector> 0.5 0.5 0.5 </bravaisVector>`

Type specifications for XML documents:

- Definition of "allowed" tags, attributes, data, and nesting in XML document (where we specify grammar)
- DTDs (document type definitions)
- XML schema - itself an XML file and W3C standard

Make XML Usable in Legacy Codes

M. Summers, et. al. (ORNL)



- Parser Generator: written once for all I/O systems
- XML Schema: developed for every class of problems
- Parser, class hierarchy, Fortran API: generated automatically