Data Management Issues in Large-Scale First-Principles Molecular Dynamics

François Gygi
University of California, Davis
fgygi@ucdavis.edu

http://www.eslab.ucdavis.edu

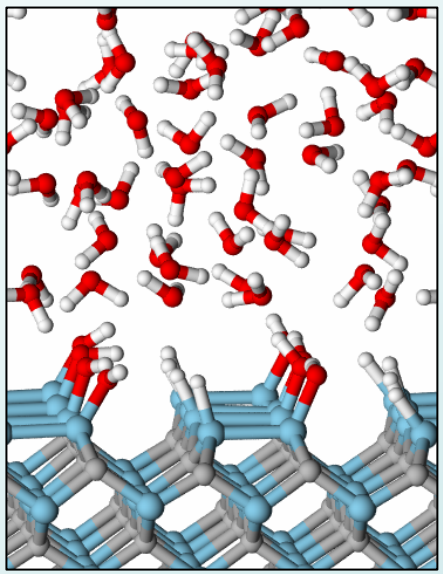
SDM Workshop, Berkeley, Dec 11, 2006

Outline

- First-Principles Molecular Dynamics: a brief introduction
- Description of the data
- Current solutions
- Future needs

First-Principles Simulations

- The goal: Simulate the properties of matter from first principles, i.e. without input from experiments
- The approach: Molecular dynamics: an atomic-scale simulation method
  - Compute the trajectories of all atoms
  - extract statistical information from the trajectories

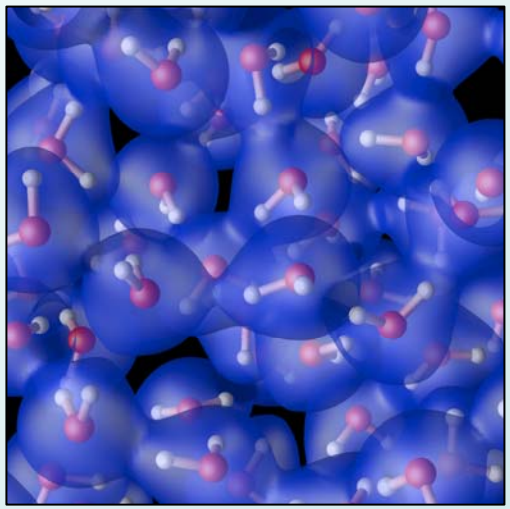


Atoms move according to Newton's law:

$$m_i \ddot{\mathbf{R}}_i = \mathbf{F}_i$$

First-Principles Simulations

- Why “First-Principles”?
  - Avoid empirical models and adjustable parameters
    - Goal: applications to situations where experimental data is not available or difficult to obtain (e.g. extreme conditions, high pressure, nanostructures, etc.)
  - Use fundamental principles: Quantum Mechanics
  - Must describe ions and electrons consistently and simultaneously

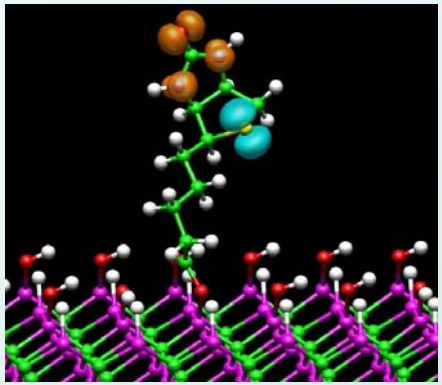


At each time step:

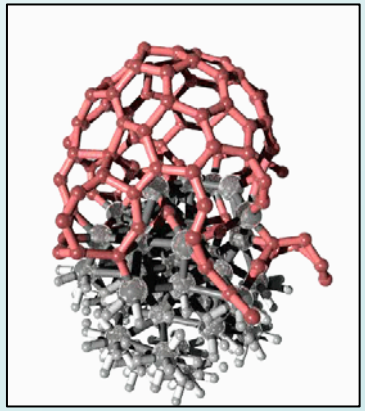
1) Compute the electronic structure
2) Derive interatomic forces
3) Move atoms

First-Principles Simulations

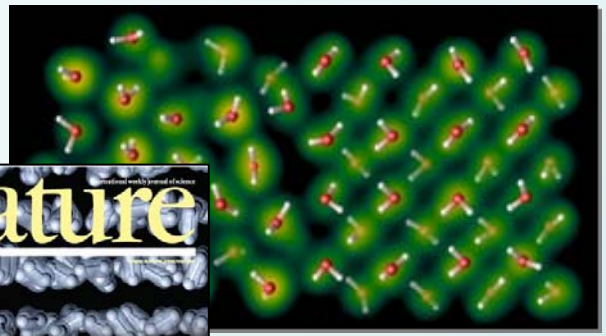
- The approach is applicable to very diverse problems
  - Chemistry
  - Nanotechnology
  - Semiconductors
  - Biochemistry
  - High-pressure physics



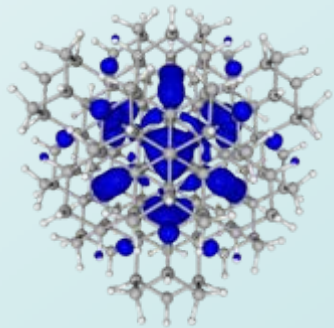
Biotin on silicon carbide



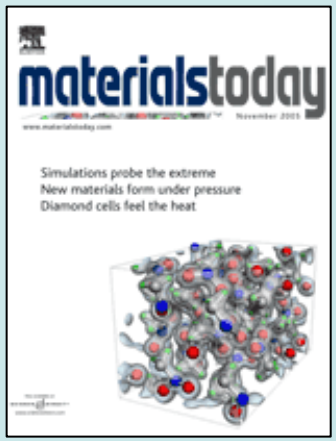
Growth of a carbon nanotube on an iron catalyst



Ice-water interface

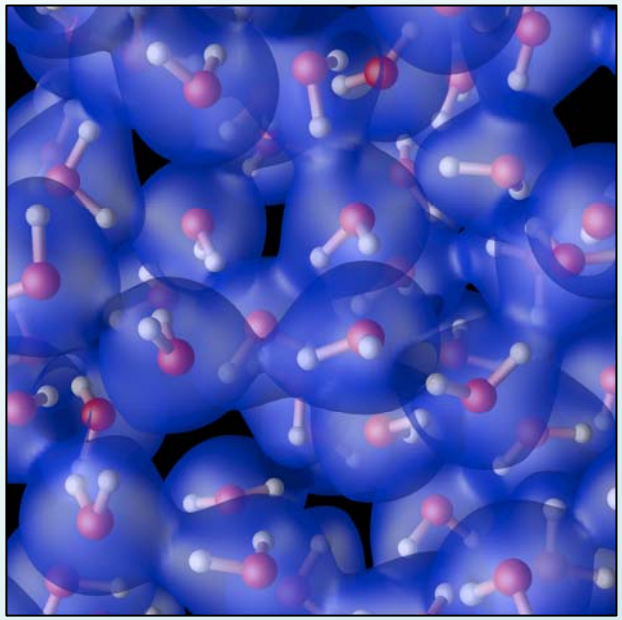


Silicon quantum dot



First-Principles Simulations

- The computation of the electronic structure is the most expensive part of the simulation, both in CPU time and memory



>99% of CPU time

At each time step:

1) Compute the electronic structure
2) Derive interatomic forces
3) Move atoms

Computing the electronic structure

- Density Functional Theory: the Kohn-Sham equations
  - solutions represent molecular orbitals (one per electron)
  - molecular orbitals are complex scalar functions in R^3
  - coupled, non-linear PDEs

$$\begin{cases} -\Delta\varphi_i + V(\rho,\mathbf{r})\varphi_i = \varepsilon_i\varphi_i \quad i = 1\ldots N_{\text{el}} \\ V(\rho,\mathbf{r}) = V_{\text{ion}}(\mathbf{r}) + \int \dfrac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,d\mathbf{r}' + V_{\text{XC}}(\rho(\mathbf{r}),\nabla\rho(\mathbf{r})) \\ \rho(\mathbf{r}) = \displaystyle\sum_{i=1}^{N_{\text{el}}} |\varphi_i(\mathbf{r})|^2 \\ \int \varphi_i^*(\mathbf{r})\,\varphi_j(\mathbf{r})\,d\mathbf{r} = \delta_{ij} \end{cases}$$

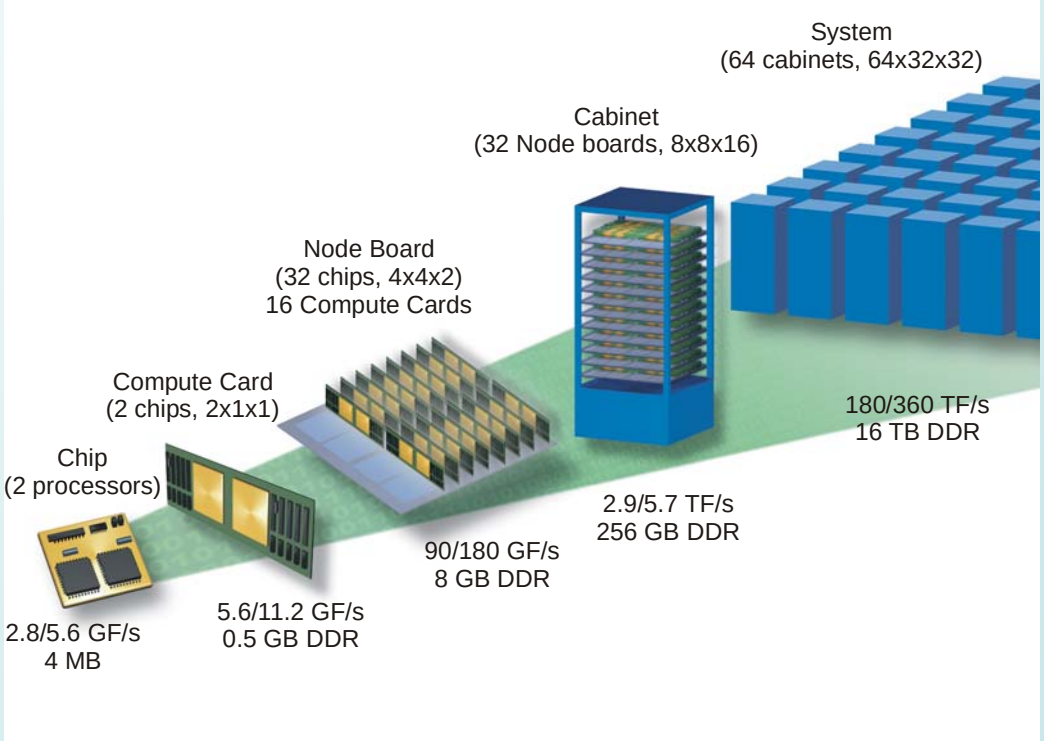
Parallel implementation of FPMD: Qbox

- Qbox is a C++/MPI implementation of First-Principles Molecular Dynamics (FPMD)
- Qbox is designed for large-scale parallel platforms and BlueGene/L
- Main design constraint: small memory footprint (< 512MB per task, or <256MB for virtual node mode)

The largest current platform: BlueGene/L

- 65,536 nodes, 128k CPUs
- 3D torus network
- 512 MB/node
- 367 TFlop peak

- Currently running on BG/Ls at
  - LLNL
  - ANL (INCITE)
  - SDSC
  - IBM T.J.Watson



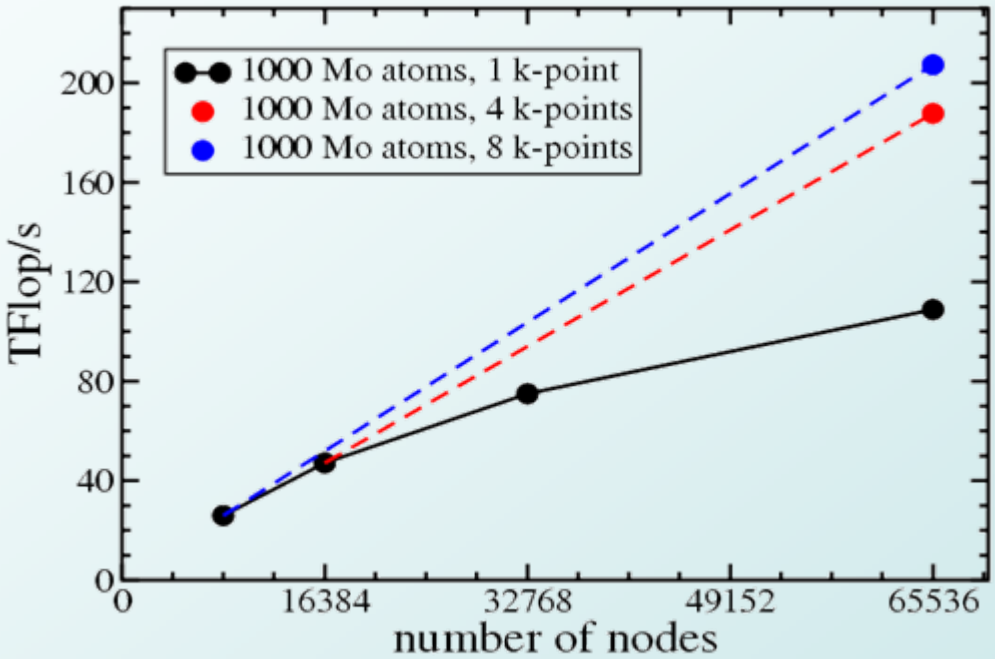
Qbox performance results on BG/L

- Simulation of a 1000-atom Molybdenum sample
- Uses 131,072 CPUs

1 k-point: 108.8 TFlop/s (30% of peak)

4 k-points: 187.7 TFlop/s (51% of peak)

8 k-points: 207.3 TFlop/s (56% of peak)



2006 Gordon Bell Award
for Peak Performance

Electronic structure data

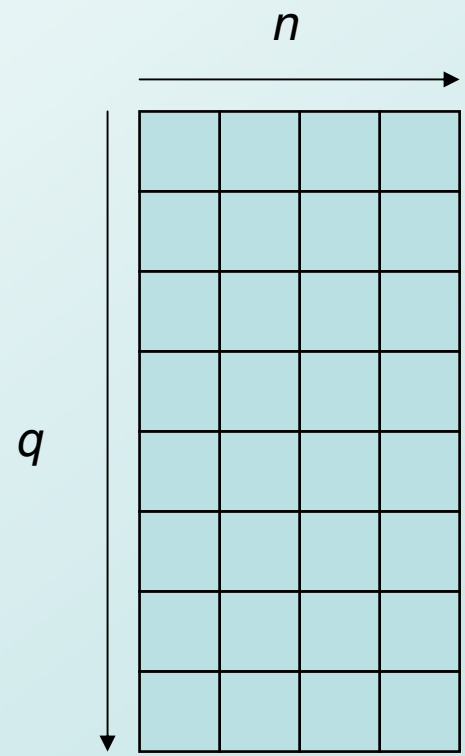
- A solution of the Kohn-Sham equations is represented by a matrix of complex Fourier coefficients c_{qn}

$$\varphi_n(\mathbf{r}) = \sum_{|\mathbf{q}|^2 < E_{\text{cut}}} c_{\mathbf{q},n} e^{i\mathbf{q}\cdot\mathbf{r}}$$

Representation of the electronic structure

- The matrix of complex coefficients c_{qn} is block distributed (ScaLAPACK data layout)

$$\varphi_n(\mathbf{r}) = \sum_{|\mathbf{q}|^2 < E_{\text{cut}}} c_{\mathbf{q},n} e^{i\mathbf{q}\cdot\mathbf{r}}$$



- Total size
  - currently: 5-50 GB
  - future: up to 1-2 TB
- One matrix represents the electronic structure at one instant

Representation of atomic positions

- Atomic positions are represented by $3N_{atoms}$ real double-precision numbers
- Total size: 50 kB at each time step for 1000 atoms
- Because of their small size, atomic trajectories are usually saved at each time step

Documents associated with FPMD

- Documents describing the simulation parameters (“input script”)
  - Qbox commands
- Documents describing atomic species (“species files”)
  - atomic number, number of valence electrons, pseudopotentials,..
- Documents describing the simulation results (“output”)
  - Energies, trajectories, etc.
- Documents describing the state of the system (“restart file”)
  - atomic positions and velocities
  - electronic wavefunctions

Where does the data reside?

- Documents describing the simulation parameters (“input script”)
  - home dir (few kB)
- Documents describing atomic species (“species files”)
  - shared file system or web server (few MB)
- Documents describing the simulation results (“output”)
  - home dir or archival system (several MB to several GB)
- Documents describing the state of the system (“restart file”)
  - parallel file system, archival system (several GB to TB)

Choice of Data Format

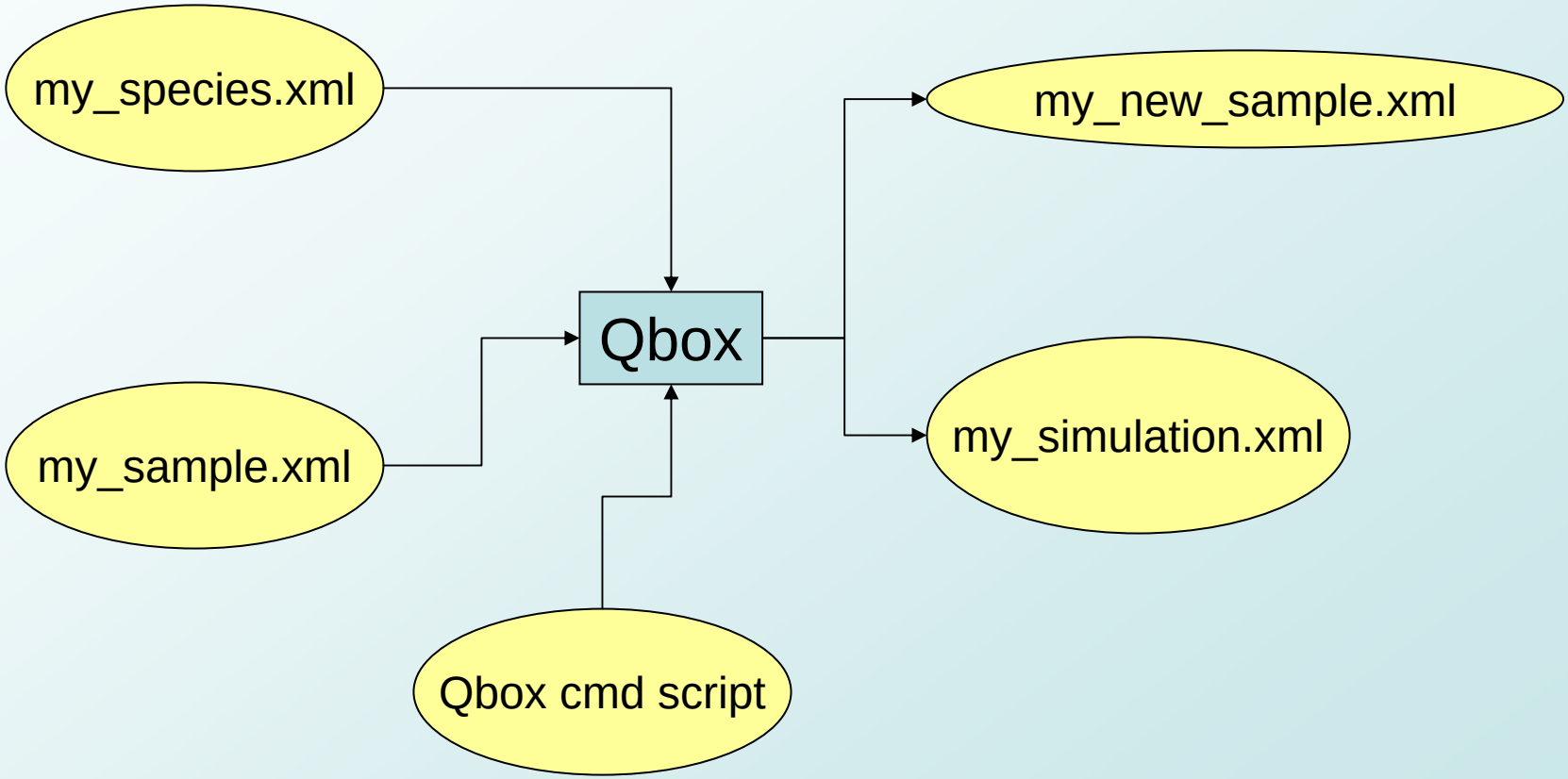
- We must be able to:
  - Exchange simulation samples with other research groups
  - Have machine-searchable datasets
  - Support validation and verification of codes
  - Develop pre-/post-processing pipelines
  - Keep track of changes in codes and in file formats

Use XML, Schemas and namespaces

Defining FPMD data standards: XML schemas

- http://www.quantum-simulation.org : web site and namespace
- **sample:** describes the state of the system
- **species:** description of an atomic species
- Other concepts under development

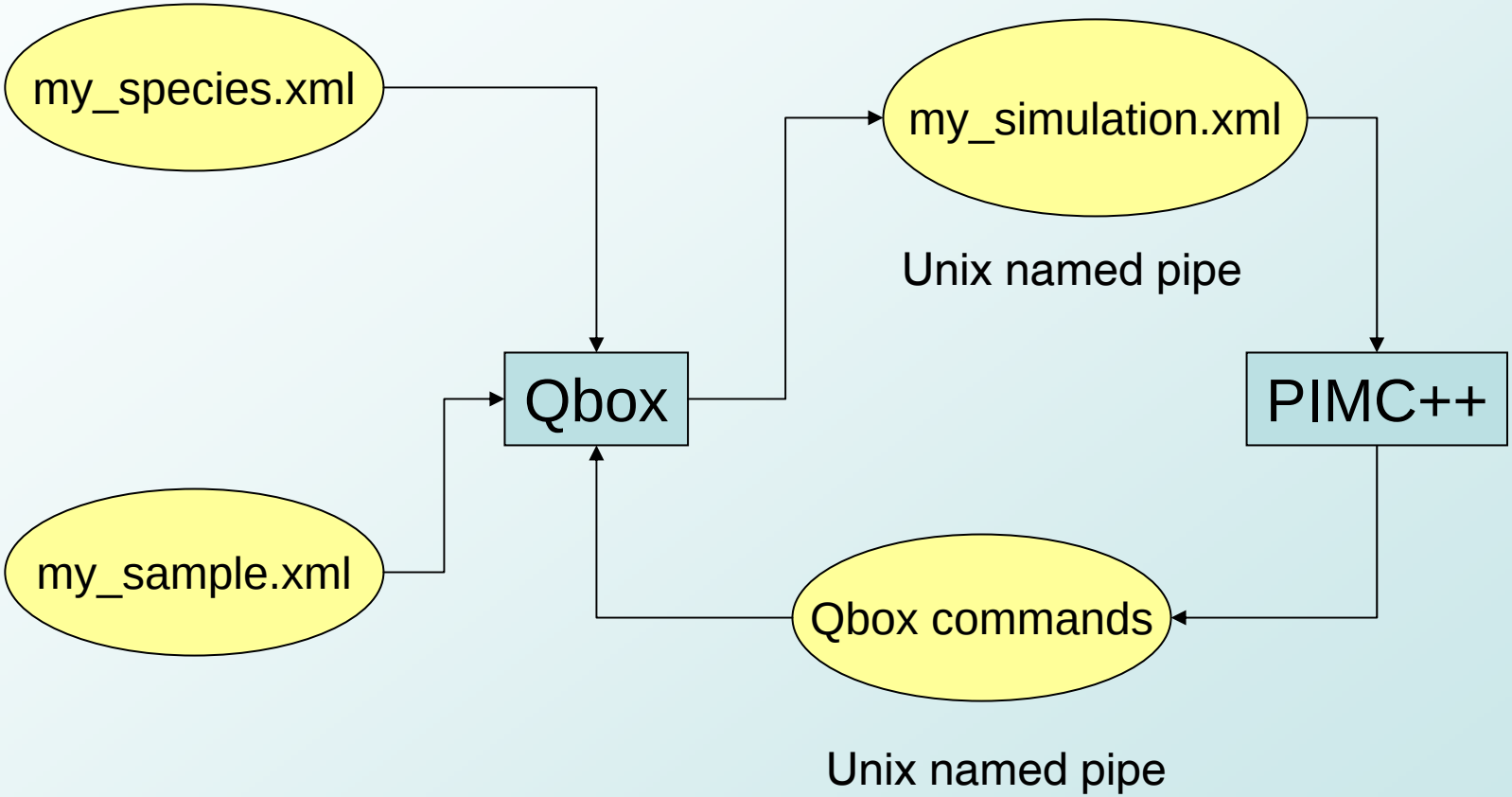
Qbox data flow



Coupling Qbox with other codes

- The results of FPMD simulations can be used as input for other, more accurate, simulations
- Example: Quantum Monte Carlo simulations (QMC)
- Two types of coupling
  - Path Integral Monte Carlo (PIMC): exchange atomic positions (small data volume)
  - Diffusion Monte Carlo: exchange electronic wavefunctions

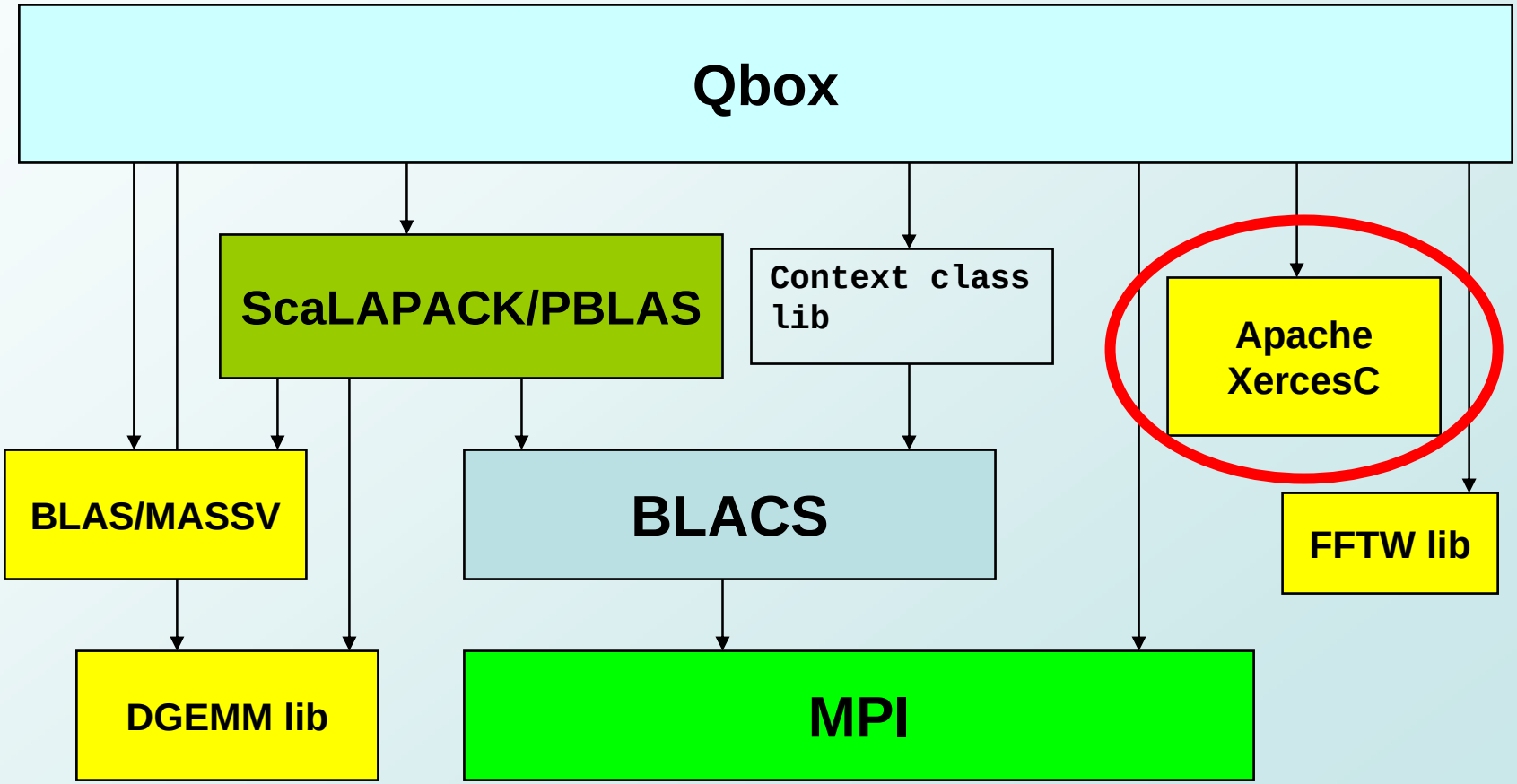
Qbox / PIMC coupling



XML Parsing in Qbox

- For each significant data object, Qbox implements ObjectReader & ObjectHandler classes
- Hierarchy of object handlers
  - an object handler can invoke another handler to parse an embedded element
  - reuse the code to read an object
- Try to keep some similarity between the C++ object model and XML document structure (but not exactly)

Qbox code structure



The Apache Xerces-C parser: useful features

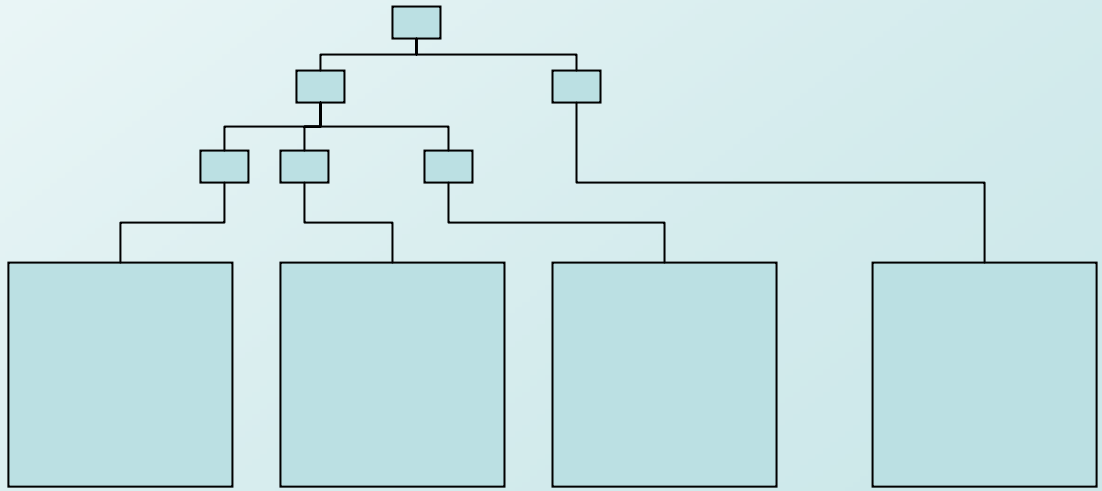
- validation
- namespace checking
- throws C++ exceptions
- supports progressive parsing
- ported to many platforms

Encoding binary data

- Sample documents contain atomic positions and electronic wavefunctions
- Wavefunction information could be saved in binary form in a separate file, but
- multiple files lead to confusion and errors: e.g. copying the parent file without copying the binary file
- Use base64 little-endian encoding
  - inflates data by 30%
  - portable
- Keep a single-file model: One sample, one file.

Parsing large files

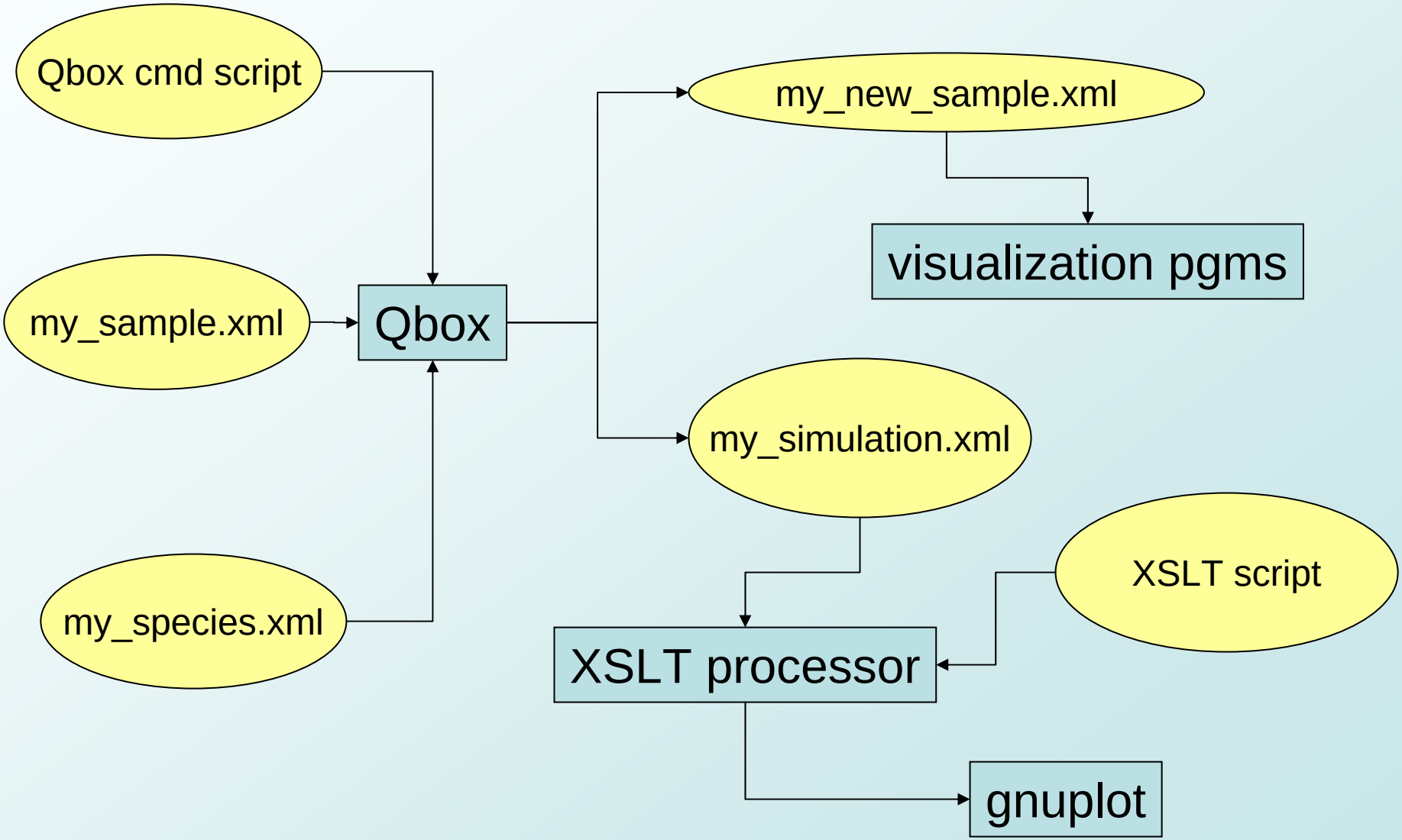
- XML samples can be large (1-50 GB)
- Sequential parsing of large XML files is slow
- Our current solution in Qbox: Parallel parsing
  - parallel parsing can be done on leaves of the document tree
  - parallel read + preprocessing of leaves
  - reduced XML document parsed in-memory by Xerces-C



Post-processing

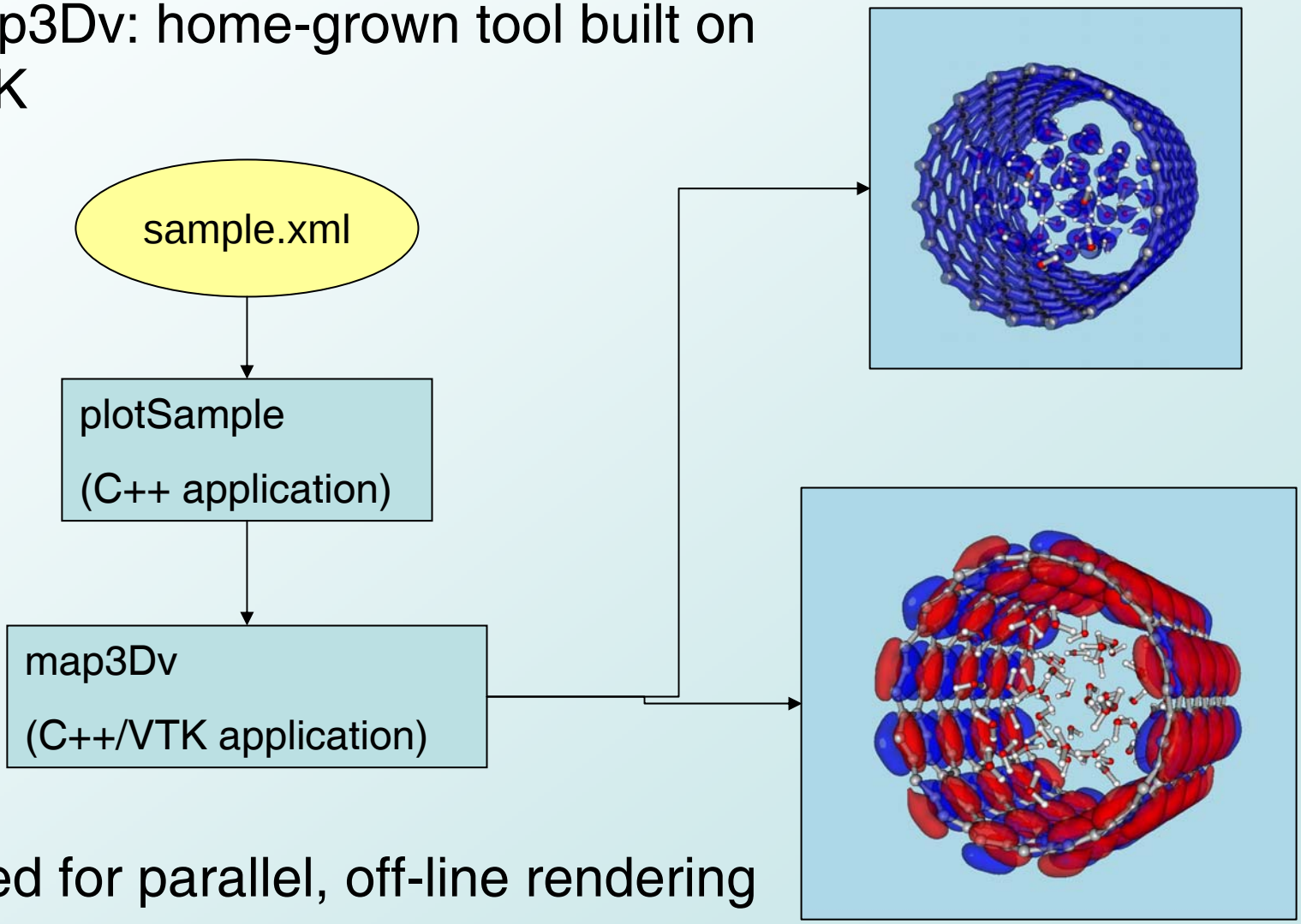
- FPMD users want to build post-processing pipelines
- Needs vary widely—no “universal” workflow
- We use the GNOME xsltproc XSLT processor
- xsltproc is namespace-aware
  - no need to track versioning throughout all post-processing pipelines
- xsltproc is web-aware
  - can post-process a web-based sample

Post-processing pipelines



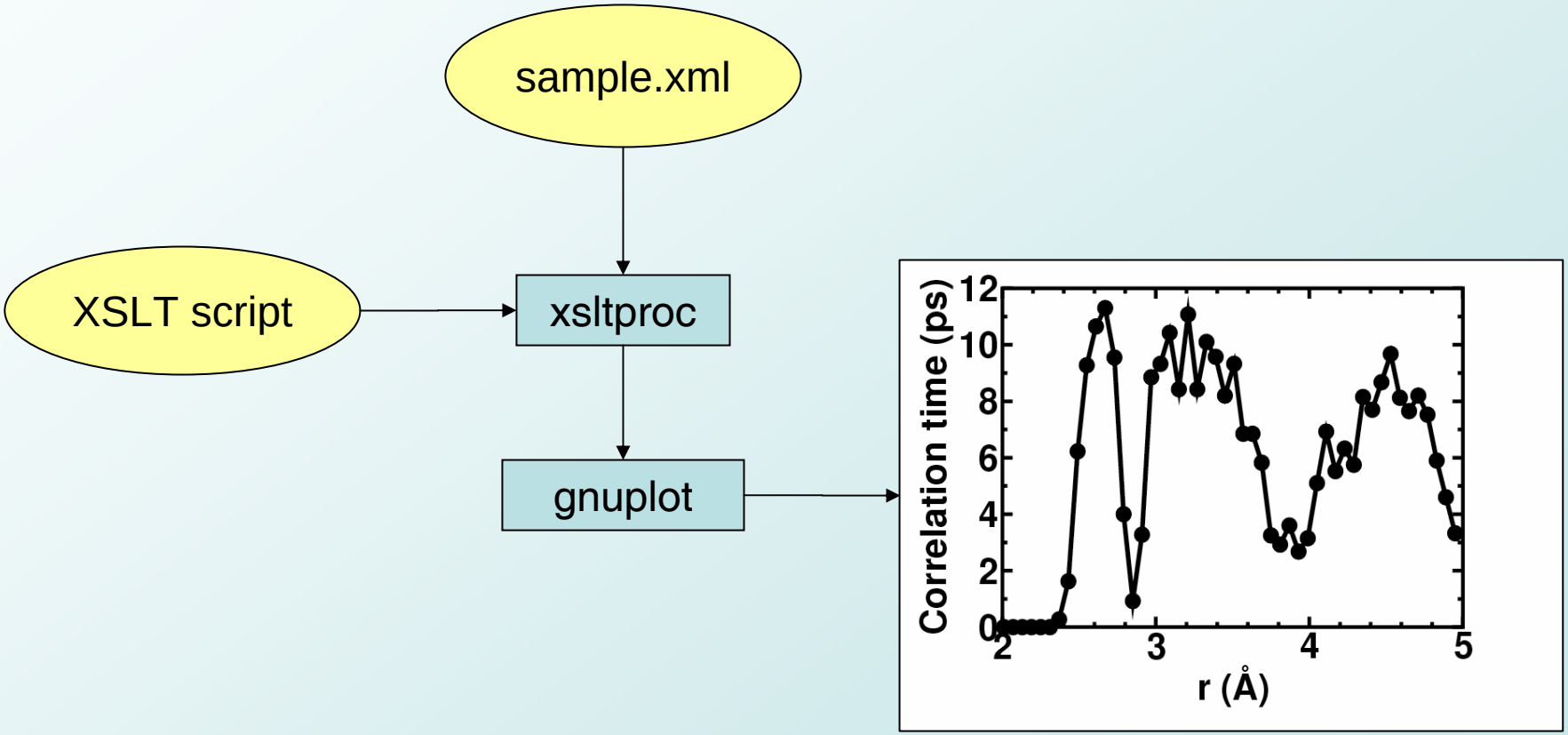
Visualization

- map3Dv: home-grown tool built on VTK



- need for parallel, off-line rendering

Data analysis



Example XSLT script

Extract the position and velocity of a given atom from a simulation

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fpmd="http://www.quantum-simulation.org/ns/fpmd/fpmd-1.0">
<!-- use: xsltproc param atomname "'O1'" test.xsl md.r -->
<xsl:param name="atomname"/>
<xsl:output method="xml" indent="yes"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/fpmd:sample">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="atomset">
  <xsl:copy-of select="atom[@name=$atomname]"/>
</xsl:template>
<xsl:template match="*"/>
</xsl:stylesheet>
```

Using web-based documents

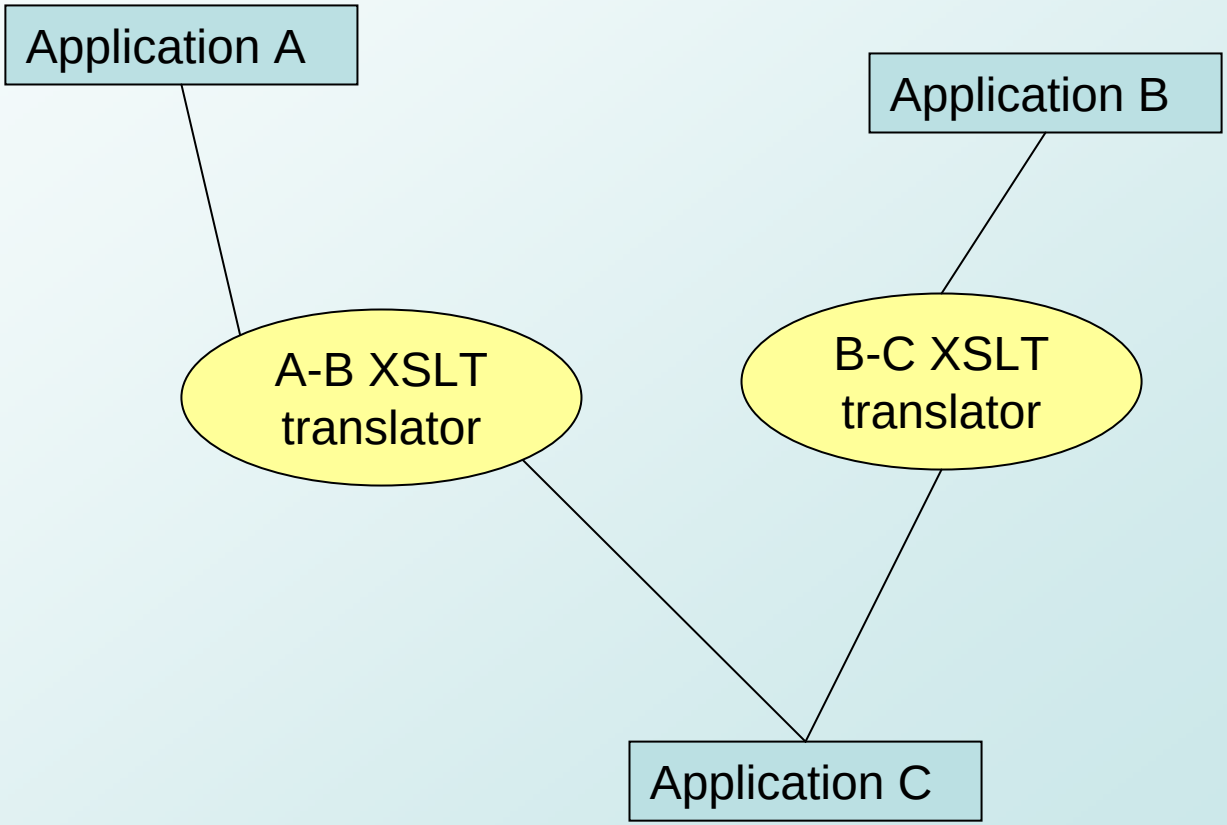
- The URI defining a sample can be a file name (e.g. mysample.xml) but also a URL (e.g. http://www.mysite.org/results/sample.xml)
- Qbox uses the Xerces-C parser
  - can use web-based pseudopotentials
  - can read web-based samples

```
<!-- [qbox] load http://www.quantum-simulation.org/examples/samples/ch4.xml -->
 <!-- LoadCmd: loading from http://www.quantum-simulation.org/examples/samples/ch4.xml -->
 <!--
 Starting XML parsing
 SpeciesHandler: found href in species definition
 name=carbon href=http://www.quantum-simulation.org/examples/species/carbon_pbe.xml
 SpeciesHandler: found href in species definition
 name=hydrogen href=http://www.quantum-simulation.org/examples/species/hydrogen_pbe.xml
 WavefunctionHandler::startElement: wavefunction nspin=1 nel=8 nempty=0
 XML parsing done
...
```

Data Compression

- Simulation data (trajectories and electronic structure) has no obvious structure or pattern
- Conventional compression algorithms are inefficient
- We develop “physics-based” compression algorithms
  - e.g. store some data at low resolution
- Trade off between space and time to recompute
  - e.g. BG/L
- New electronic structure methods use O(NlogN) data instead of N^2.

Long-term goal: interoperable XML applications



Summary

- Scaling First-Principles Molecular Dynamics to petaflop platforms will require
  - efficient parallel file systems
  - fast, parallel XML parsers
  - tools to move data to other sites
- Post-processing pipelines
  - fast XSLT processors
- Reduction of data volume
  - new “physics-based” compression algorithms
  - new “linear-scaling” electronic structure methods