Multi-threaded Frameworks Session

Development of multi-threaded HEP data processing frameworks

Session description

- This is a technical session with concentration on work that is specific to overall frameworks and moving them towards multi-core, many-core, or perhaps other high-performance parallel systems. Areas of interest are the development of multi-threaded HEP data processing frameworks, design issues such as scheduling, thread-safety of toolkits, and performance estimation. We will also discuss migration strategies for the existing algorithmic code and lessons learned from prototypes and demonstrators.

Schedule

- Development of software frameworks
  - CMSSW - Chris Jones
  - art framework at the Intensity Frontier - Chris Green
  - The CERN Concurrency Framework Project (CF4Hep) - Benedikt Hegner
  - Simulation vector prototype - Andrei Gheata
- Uses of software frameworks
  - ATLAS multithreading experiences - Wim Larijsen
  - Intra-module parallelism - Danilo Piparo
  - Trigger integration tools - Kurt Biery
  - Parallelism of algorithms with TBB in the SuperB framework - Marco Corvo
- Discussion

Things to be addressed in the talks

- Summary of technical issues
  - Have you decided on a technology?
  - Are there valuable insights you've gained or "lesson's learned" from the last year?
- Scheduling of work
  - At what level? (Event level, module level, other)
  - How will processing be handled? (Control-flow or demand-driven)
  - What level of parallelism will be achieved? (event and/ or module level, multi-node or multi-process)

Things to be addressed in the talks (2)

- Component design
  - Are there any changes to global Service? (addressing bottlenecks, shared access issues)
  - Is the Event data model changing? (product design, access rules, metadata)
  - Is geometry, calibration or other ancillary data handling changing
  - How will summary products such as histograms and statistics be handled?
  - Is there a need for parallelism within the I/O subsystem?

Things to be addressed in the talks (3)

- Programming and algorithm integration
  - How will modules/algorithms be organized? (interfaces, granularity and separation)
  - What guidelines for developing classes and functions will be developed?
  - How will existing code be migrated or ported?
  - How will resources for parallel algorithms be managed? (memory, cores, etc.)
- Performance issues
  - Are there results from the past year from demonstrators and various use cases?
  - What measures of effectiveness and success have been developed?
  - What scaling studies were performed and how many cores are you targeting?

Reminder - Objectives from last year

- Assess the interest for a joint development of a generic software framework or a collection of services.
- Determine the degree of overlap between the needs and medium to long term plans of the experiments Fermilab and CERN are supporting.
- Identify a list of potential technologies that would allow the effective use of upcoming hardware landscape and ensure that all the relevant solutions are being explored.
- Identify commonalities and synergies that lend themselves to collaboration.

Session begins …

Technology: Are there clear winners?

- Promising programming tools for C++
  - TBB
  - OpenMP
  - GCD / libdispatch
- Other technologies
  - Go
  - pthreads directly
  - has C++11 made things much easier to deal with?
- Architectures
  - GPGPU with OpenCL or CUDA
  - Phi
  - Anything else? Blue Gene?

Technique: What is most advantageous?

- Programming methodologies
  - Whiteboard
  - "vectorizing"
  - Task parallelism
  - Data parallelism
  - Structure of arrays versus array of structures
- Levels
  - Many events at once
  - Many modules within one event
  - Within modules - in user space
- Performance gains