CHEP 2009
Prague

Multi-threaded Event Reconstruction with JANA

JANA

JANA is a multithreaded event reconstruction framework written in C++. It is designed to utilize all of the available cores of a CPU while processing high volume HENP data.

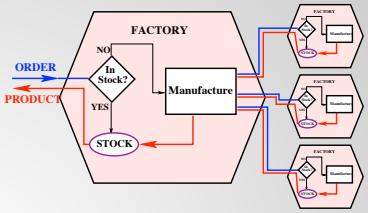
Website : http://www.jlab.org/JANA

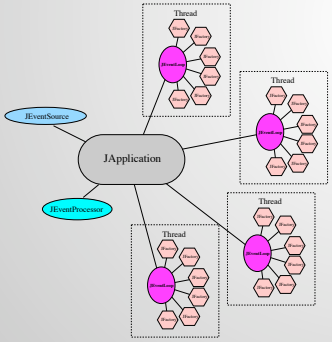
Data On Demand

JANA's modified factory model produces data only on demand. This avoids wasting CPU cycles on reconstruction that is not needed for that particular event.

Alternate Factory Model

Reconstruction code is built into factory classes as callback methods. Inputs come from other factories. In JANA's model, ownership of the objects stays with the factory. Only const pointers are passed, ensuring data integrity.



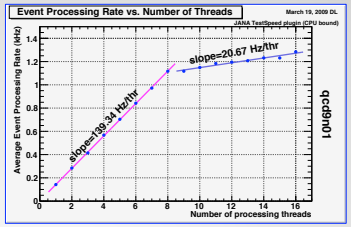


Threading Model

A JANA application consists of a single *JApplication* object and multiple *JEventLoop* objects (one for each thread). Each thread has its own complete set of factory objects that together, can completely process an event. JANA uses POSIX pthreads.

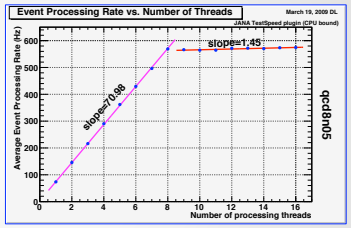
CPU bound Jobs

CPU bound jobs benefit from the parallelism achieved through multi-threading. This benefit can be lost if the framework requires frequent mutex (un)locking. These plots show event processing rates for CPU bound jobs. The linear shape indicates proper scaling with the number of threads which implies virtually no overhead is imposed by the framework which is designed to minimize mutex (un)locking.



Intel Xeon (5560)
2.8GHz
Dual Processors
8 cores/processor + hyperthreading

For this test, each hyperthread gave the equivalent of 15% of a full core

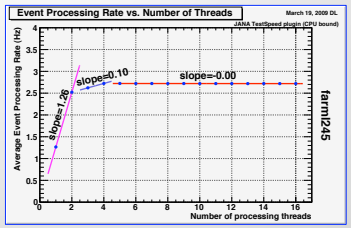


AMD Opteron (2352)
2.1GHz
Dual Processors
4 cores/processor with ***no*** hyperthreading

Without hyperthreading, a modest improvement (2% of a core) is observed when processing threads are over-subscribed

Hyperthreading

These plots also show how "hyperthreading" in Intel CPUs can improve performance for CPU bound jobs

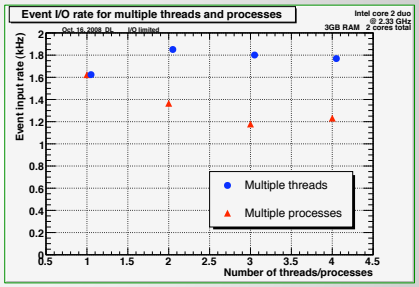


Intel Xeon (circa 2004)
2.8GHz
Dual Processors with 1 core/processor + hyperthreading

An older machine shows hyperthreads gaining only about 8% of a core.

I/O bound Jobs

Multiple processes accessing the same disc leads to competition for the position of the read head. A multi-threaded process can stream events in sequence, dispatching them to individual threads resulting in faster event processing rates for I/O bound jobs as shown in these test results.



DAVID LAWRENCE davidl@jlab.org