Computing R&D activities: Framework

Marco Corvo

CNRS and INFN

I SuperB Collaboration Meeting

September 15, 2011

Group: The people

- The group is currently made of five people: Francesco Giacomini, Vincenzo Ciaschini and Paolo Franchini, joining in a couple of weeks (CNAF), me (CNRS & INFN Padova) and Stefano Longo (INFN Padova)
- We started a couple of months ago a discussion on how to rethink the current SuperB framework, which is essentially the one inherited from BaBar
- We can also count on the “moral” support of Peter Elmer and Vincenzo Innocente, who agreed to help us find out the right path in our quest for the best solutions, both in terms of architecture and implementation

(Very) Short term plans

- Tools:
    - Definition of standards: adopt the new C++11 definitions and gcc 4.6
    - Use of CMake as build system and Git as repository for the 'exercises'
        - This has a double goal: keep things clean and improve our knowledge/experience of these systems, which can be adopted SuperB-wide as collaborative tools

(Very) Short term plans

- Goals:
  - We need to acquire some experience with the old system in order to understand its mechanisms
    - Isolate the current framework, build it as a standalone tool and use it to execute some exercises ('Hello World' to start with)
  - Parallelization exercises: Analysis level
    - Extend the exercises trying to define a (simple) graph of dependencies among some (possibly few) physics modules in order to exploit parallel scheduling
    - Evaluate some products: Intel Thread Building Blocks (TBB), Apple Grand Central Dispatch (GCD), Boost.Task . . .
  - Parallelization exercise: Algorithm level
    - Recostruction, TrackFinding, others?

Next meetings and actions

- Francesco will put in our Wiki page the receipt to build the current framework as a standalone package
- Padova will help on the setup (CMake build plus Git repo)
- Within some days we'll have a MIC based box (40 cores, 80 with hyperthreading on) ready for performance measures on some SuperB executables (SkimMini, PacUserApp)
- With these collected data Vincenzo will produce a further report based on the tests already made (CPU, memory, I/O) plus MIC based measures
    - Some measurements are reported later in this talk
- Soon after this Collaboration Meeting we will meet again to define operationally the real exercises to start with.

The goal

- Last July Vincenzo (who provided these slides) presented data about performance analysis of PacUserApp and SkimMiniApp.
- There was an unexplained case of contention for PacUserApp
- What follows is an analysys of that contention.

The goal

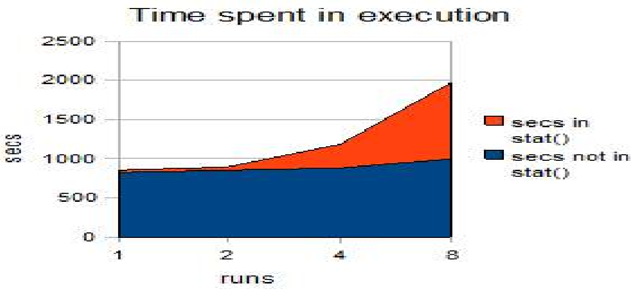
- Last July Vincenzo (who provided these slides) presented data about performance analysis of PacUserApp and SkimMiniApp.
- There was an unexplained case of contention for PacUserApp
- What follows is an analysys of that contention.

The measures

- Run SkimMiniApp under “strace -tT -f” to see where time is spent.
- Pipe output through “| gawk 'print strftime(), $0” to be able to correlate with strace output.

Causes

- stat()!
- The SkimMiniApp executable calls stat() 961890 times!
- Of these, 937304 are done before the evaluation of the second event starts.
- They do go into contention, going from taking around 0.00001 secs to 0.0003 or more each!



Conclusions

- It is clear that stat() is the main cause of contention
- Most of the calls happen before the evaluation of the second event
- It implies that this cost is less and less evident the longer the run is
    - Which is probably why it was unnoticed before
- There is still a little delay.
    - Test harness will use some resources, though