Strategies for CPU Optimization

Paolo Calafiura

The Economics of CPU Optimization

- It takes several thousand functions to account for 80% of reconstruction CPU usage (D. Levinthal)
- Developers time is more valuable than CPU time

yes, but

- ATLAS today uses 66K cores (M. Ernst) at a cost of 200-500 CHF/core/year (Y. Yao)

1% CPU gain >100K CHF/year savings

Spending a week to gain 0.1% CPU
very profitable use of your time!

Attacking the Long Tail

A Traditional Approach

- Gain 10% CPU by cutting CPU usage of top 10 functions by 50%
  - Analyze function logic, look for alternative algorithms
- Gain 10% CPU by cutting CPU usage on next 100 function by 25%
  - Focus on technical problems
- Crucial to have access to line-by line profiling as guide
  - Valgrind too slow, alternatives?

Call-driven Optimization

A somewhat orthogonal approach

- Function calls cost time, inhibit optimization, slow down processor
- Clusters of functions called many times seem to be an indicator of optimization opportunities (Roberto, Rolf)

Waiting for Superman, or not...

- Compiler developers get it:
  more aggressive inlining is a big focus of their work BUT
- General solutions are hard to develop
  - First serious attempts in gcc 4.6, at least one year away for us
- We need to help
  - Declare inline anything that may make sense to inline
    - Compiler may only inline what is declared inline
      - Won't inline everything that is declared inline
  - Devirtualize methods, either using “final” or replacing interfaces with templates (remember the INav4Mom debate?)
  - Ruthlessly sacrifice generality to performance, remember

1% CPU > 100KCHF/year