

---

# National Energy Research Scientific Computing Center (NERSC)

## Hephaestus – A Memory Tracker

Wim T.L.P. Lavrijsen

NERSC HENPC, LBNL

LBL Software Meeting - 05/11/06





# Context

- **Atlas closing in on start-up**
  - Code to be brought to production quality
    - CPU & memory profiling
    - Memory leak detection
    - Address checking
    - Call-graphs understanding
    - General debugging
  - Constraints
    - Atlas software is super-sized
    - Many programmers whose codes interact



# Current tools (1)

- **Valgrind (KDE)**
  - Emulates CPU execution
    - Exhaustive, extensive, detailed
    - Memory hungry, and extremely slow
  - Since recently, in-process
    - Limited execution model, easier to crash
- **Oprofile (Levon, Elie / SourceForge)**
  - Statistics based profiler (reads CPU regs)
  - Kernel module (i.e. needs root access)
  - Requires debug symbols available



## Current tools (2)

- **Memprof (RedHat)**
  - Requires Gnome (GUI), pre-loading
  - From FAQ:  
Is there any documentation?  
Not currently, no. Contributions would be appreciated.
- **PerfTools (Google)**
  - Requires relinking, crash-prone
  - Initially not much, but still very active
    - Track on-going development



# Issues

- **Tools written by hackers for hackers**
  - Use, reports, documentation (if exists)
    - Black magic to most of our developers
  - Not in line with Ixplus reality
    - Multi-user, limited resources, CERN IT
  - Not in line with Atlas build system
    - Re-compilation, re-linking are no-no's
- **Atlas software specific issues**
  - Size; existing sighandlers and preloads



# Why do we care?

- **User finds GroovyTool on the web**
  - Reads example: “just press this button”
- **Applies tool on Atlas software**
  - SegFault / bad\_alloc / no report
- **Starts blaming Athena and/or python**
- **Bludgeon developer into doing the work**
  - Little wiser: 100MB report full of LCG
- **Repeat process ... per user / per release**
  - Heavily bottlenecked, wasteful process



# To our advantage

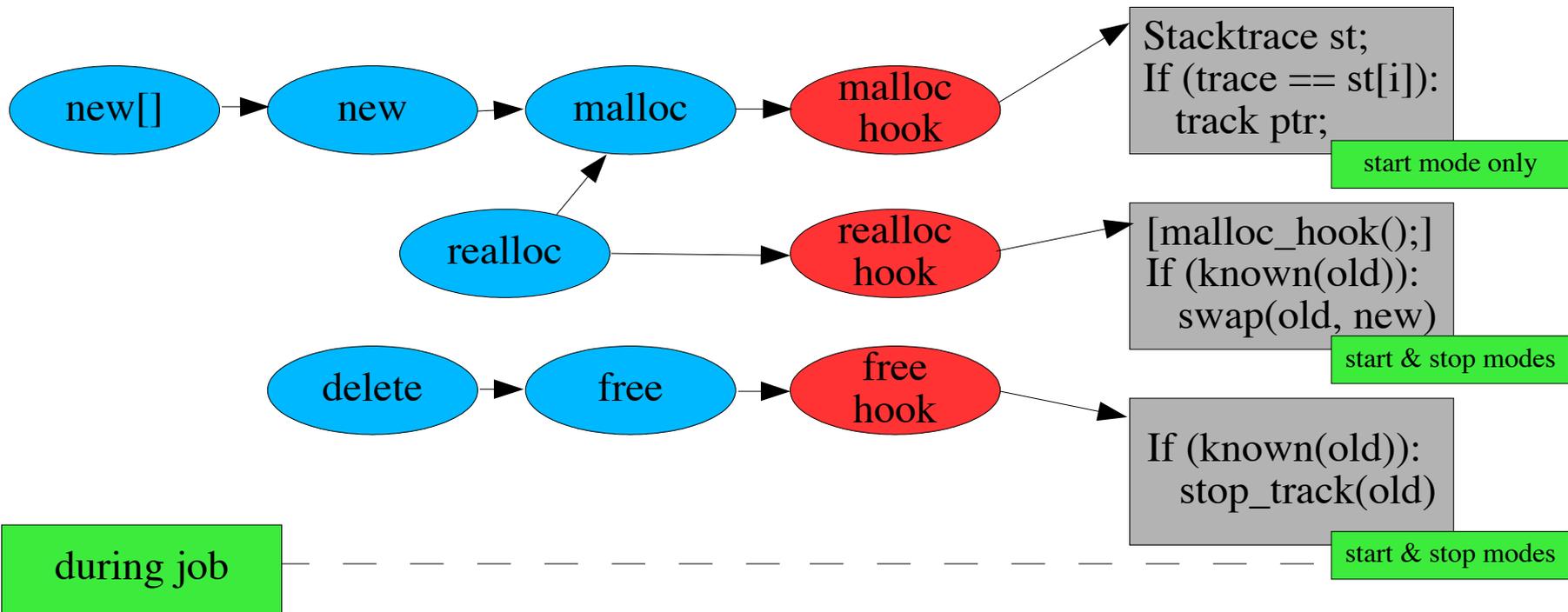
- **Platform: gcc > v3**
  - Linux (and MacOS X)
  - abi, glibc, gdb, bintils
- **Python interpreter loads Athena**
  - Controlled environment
  - A “before” and “after” Athena
    - Code before creating ApplicationMgr
    - Code after calling Terminate()
  - *Note: oversimplification (preloads)*



# MemoryTracker idea

- **Use glibc malloc\_hooks**
  - Easy to write if they didn't already exist
- **Load memory tracker into python**
  - Make tracker tuned to Athena needs
    - E.g. focus on Algorithms in execute phase
  - Settings can be made in job options
    - Or, equivalently, on interactive prompt
  - Install/uninstall, start/stop of hooks
- **Report after Athena Terminate()**
  - Process still fully alive (if needed)

# MemoryTracker



for all ptrs still tracked:  
 create unique list  
 resolve stack trace  
 if ( ! filter passed )  
 continue  
 print report

at job end



# Technical issues

- **STL allocators use mutexes**
  - Can't use default ones within hooks
  - Implement basic malloc allocator
    - And code so as to prevent re-entrance
- **Stack traces are slow, large in size**
  - Track addresses instead
  - Cache lookups b/c of possible relocation
- **Stack may be too deep for filter**
  - Detect, report, reconfigure (TBD)



# Results, example

detected 1367 non-unique leaks in "Algorithm::sysExecute"

...

leak detected, originating in: CaloClusterBuilderSE::CreateImpactInCalo  
(Trk::Track const\*) (28 bytes)

0x8272ef2 CaloClusterBuilderSE::CreateImpactInCalo(Trk::Track const\*)

0x82727e2 CaloClusterBuilderSE::execute(Rec::TrackParticle const\*,  
EMTrackMatch\*)

0x833160a softBuilder::execute()

0x5e7e41 Algorithm::sysExecute()

0x22af5f AthenaEventLoopMgr::executeAlgorithms()

0x22b3ae AthenaEventLoopMgr::executeEvent(void\*)

0x22bd42 AthenaEventLoopMgr::nextEvent(int)

0x11b8c53 MinimalEventLoopMgr::executeRun(int)

...

ignored 210 unique leaks (used 1 filter)



# Conclusions

- **Tool now in Atlas nightly builds**
  - Use `--leak-check-execute` on athena CLI
  - Waiting for feedback from developers
- **Many leaks detected in reconstruction**
  - Several of them trivial => proof it works
  - Only tool to leak-check full Atlas reco
- **Looks successful, mem profile next**
  - Focussing on Athena initialize()