fads
a (Go-based) FAst Detector Simulation toolkit

Sébastien Binet

LAL/IN2P3

2015-01-21

fads

fads is a "FAst Detector Simulation" toolkit.

- morally a translation of C++-Delphes into Go
- a testbed for R&D in Go and concurrent frameworks
- uses go-hep/fwk to expose, manage and harness concurrency into the usual HEP event loop (initialize | process-events | finalize)

Code is on github (BSD-3):

- https://github.com/go-hep/fwk
- https://github.com/go-hep/fads

Documentation is served by godoc.org, Continuous Integration by drone.io

- https://godoc.org/github.com/go-hep/fwk
- https://godoc.org/github.com/go-hep/fads

go-hep/fads - Installation

As easy as:

```
$ export GOPATH=$HOME/dev/gocode
$ export PATH=$GOPATH/bin:$PATH

$ go get github.com/go-hep/fads/...
```

Yes, with the ellipsis at the end, to also install sub-packages.

- `go get` will recursively download and install all the packages that `go-hep/fads` depends on. (no `Makefile` needed)
- you get a statically linked executable in a matter of seconds (even for large projects)
- simple deployment and distribution
- the speed of development of `python` with the speed of execution of `C++`

go-hep/fwk - Concurrency

go-hep/fwk enables:

- event-level concurrency
- tasks-level concurrency

go-hep/fwk relies on Go's runtime to properly schedule goroutines.

For sub-task concurrency, users are by construction required to use Go's constructs (goroutines and channels) so everything is consistent and the runtime has the complete picture.

Note: Go's runtime isn't yet NUMA-aware.
A proposal for Go-1.5 (June-2015) is in the works

go-hep/fads - real world use case

- translated C++-Delphes' ATLAS data-card into Go
- go-hep/fads-app
- installation:

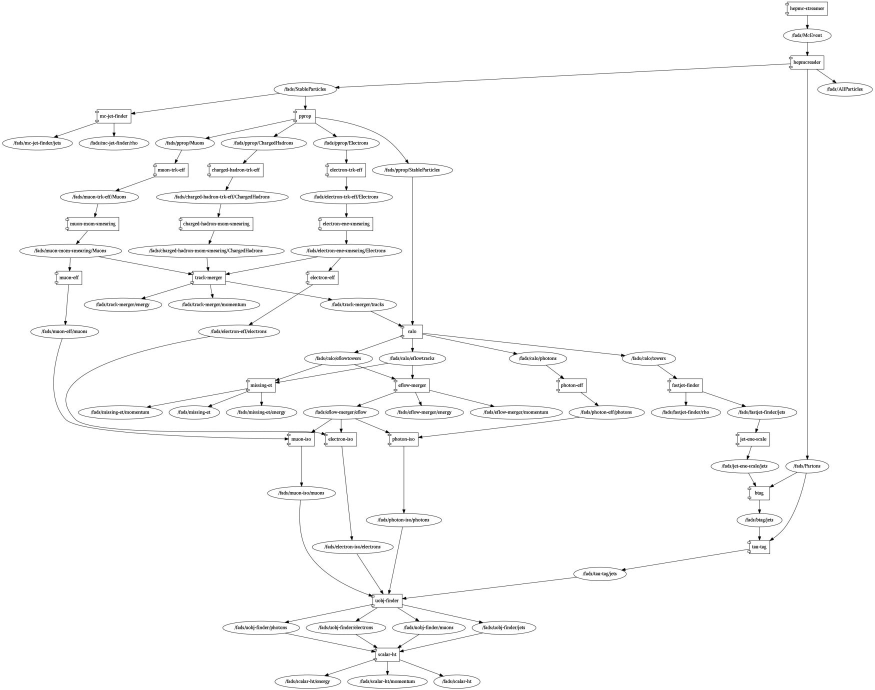
```
$ go get github.com/go-hep/fads/cmd/fads-app
$ fads-app -help
Usage: fads-app [options] <hepmc-input-file>

ex:
 $ fads-app -l=INFO -evtmax=-1 ./testdata/hepmc.data

options:
  -cpu-prof=false: enable CPU profiling
  -evtmax=-1: number of events to process
  -l="INFO": log level (DEBUG|INFO|WARN|ERROR)
  -nprocs=0: number of concurrent events to process
```

go-hep/fads - components

- a HepMC converter
- particle propagator
- calorimeter simulator
- energy rescaler, momentum smearer
- isolation
- b-tagging, tau-tagging
- jet-finder (reimplementation of FastJet in Go: go-hep/fastjet)
- histogram service (from go-hep/fwk)

Caveats:

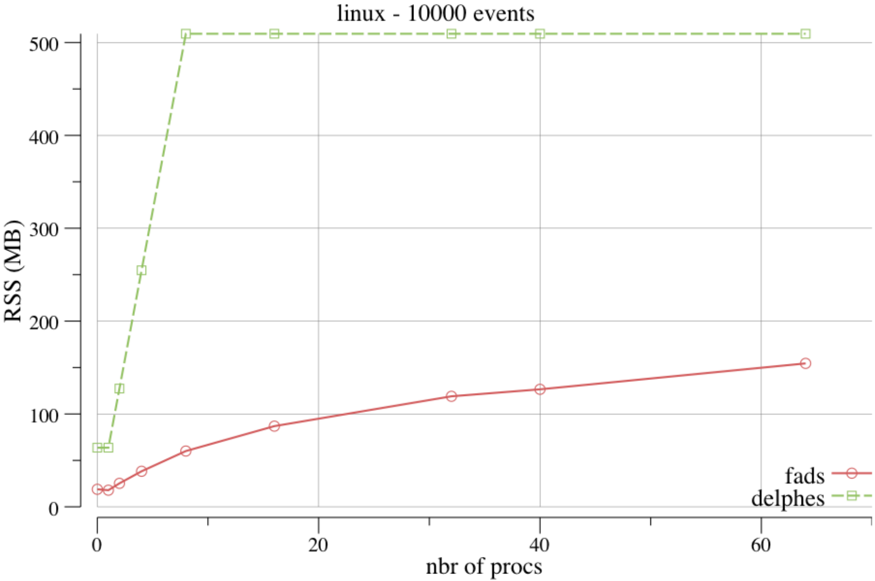
- no real persistency to speak of (*ie:* JSON, ASCII and Gob)
- jet clustering limited to N^3 (slowest and dumbest scheme of C++-FastJet)


hepmc-streamer
/fads/McEvent
hepmcreader
/fads/StableParticles
/fads/AllParticles
mc-jet-finder
pprop
/fads/mc-jet-finder/jets
/fads/mc-jet-finder/rho
/fads/pprop/Muons
/fads/pprop/ChargedHadrons
/fads/pprop/Electrons
/fads/pprop/StableParticles
muon-trk-eff
charged-hadron-trk-eff
electron-trk-eff
/fads/muon-trk-eff/Muons
/fads/charged-hadron-trk-eff/ChargedHadrons
/fads/electron-trk-eff/Electrons
muon-mom-smearing
charged-hadron-mom-smearing
electron-ene-smearing
/fads/muon-mom-smearing/Muons
/fads/charged-hadron-mom-smearing/ChargedHadrons
/fads/electron-ene-smearing/Electrons
muon-eff
track-merger
electron-eff
/fads/track-merger/energy
/fads/track-merger/momentum
/fads/track-merger/tracks
/fads/muon-eff/muons
/fads/electron-eff/electrons
calo
/fads/calo/eflowtowers
/fads/calo/eflowtracks
/fads/calo/photons
/fads/calo/towers
missing-et
eflow-merger
photon-eff
fastjet-finder
/fads/missing-et/momentum
/fads/missing-et
/fads/missing-et/energy
/fads/eflow-merger/eflow
/fads/eflow-merger/energy
/fads/eflow-merger/momentum
/fads/photon-eff/photons
/fads/fastjet-finder/rho
/fads/fastjet-finder/jets
muon-iso
electron-iso
photon-iso
jet-ene-scale
/fads/jet-ene-scale/jets
/fads/Partons
btag
/fads/muon-iso/muons
/fads/photon-iso/photons
/fads/btag/jets
/fads/electron-iso/electrons
tau-tag
/fads/tau-tag/jets
uobj-finder
/fads/uobj-finder/photons
/fads/uobj-finder/electrons
/fads/uobj-finder/muons
/fads/uobj-finder/jets
scalar-ht
/fads/scalar-ht/energy
/fads/scalar-ht/momentum
/fads/scalar-ht

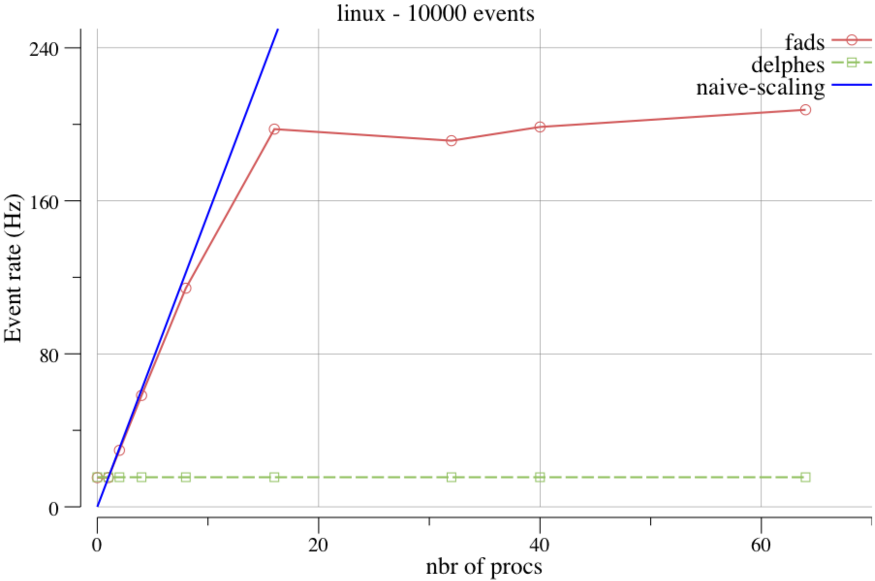
Performances - testbenches

- Linux: Intel(R) Core(TM)2 Duo CPU @ 2.53GHz, 4GB RAM, 2 cores
- MacOSX-10.6: Intel(R) Xeon(R) CPU @ 2.27GHz, 172GB RAM, 16 cores
- Linux: Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz, 40 cores

Linux (40 cores) testbench: memory



Linux (40 cores) testbench: event throughput



go-hep

go-hep project

A set of pure-Go or bindings to HEP libraries

- go-hep/fads: fast detector simulation toolkit
- go-hep/fastjet: jet clustering algorithms (*WIP*)
- go-hep/fmom: 4-vectors
- go-hep/fwk: concurrent framework
- go-hep/hbook: histograms and n-tuples (*WIP*)
- go-hep/hplot: interactive plotting (*WIP*)
- go-hep/hepmc: HepMC in Go (EDM + I/O)

go-hep + astrogo projects

- go-hep/hepevt: HEPEVT bindings
- go-hep/heppdt: HEP particle data table
- go-hep/lhef: Les Houches Event File format
- go-hep/croot: bindings to a subset of ROOT I/O
- go-hep/rio: go-hep record oriented I/O
- go-hep/sio: LCIO I/O
- go-hep/slha: SUSY Les Houches Accord I/O

- astrogo/cfitsio: bindings to FITSIO
- astrogo/fitsio: pure Go I/O for FITS files
- astrogo/vo/votable: I/O for VOTable (*WIP*)

- sbinet/hdf5: bindings to HDF5

go-hep & HSF

Most of development workflow already addressed (doc, CI, DVCS)
HSF could provide (from go-hep POV):

- wider audience (users, developers)
- test machines/architectures
- storage area for input data (for tests) and/or (binary) releases
- agreement on cross-language interoperability (file formats, data layout (POD)) in a non pure-C++ environment