TECHNISCHE UNIVERSITÄT DRESDEN
Center for Information Services and High Performance Computing (ZIH)

The Benefit of GCC's open structure on instrumentation in the HPC area

9. Juli 2012

Johannes Ziegenbalg, Bert Wesarg

ZIH
Center for Information Services & High Performance Computing

Table of contents

Intstrumentation

Definition:

”In context of computer programming, instrumentation means to enrich the source code of a program with additional information.”

(acc. de.wikipedia.org)

Our approach

enrich source code:

- insert function call at beginning & end of function
- calls to ”instrumentation functions”

additional information:

- assign id of function
- collect list of function metadata (name, scl...)
- time-stamp of execution-point

The GCC approach

enrich source code:

- implementation of:
  `__cyg_profile_func_enter(...)`
  `__cyg_profile_func_exit(...)`
- compile with: *-finstrument-functions*

additional information:

- you get address of function
- function metadata has to be computed from symbol table

Tracing

Tracing

Definition:

”In software engineering, tracing is a specialized use of logging to record information about a program’s execution.”

(en.wikipedia.org)

Our approach

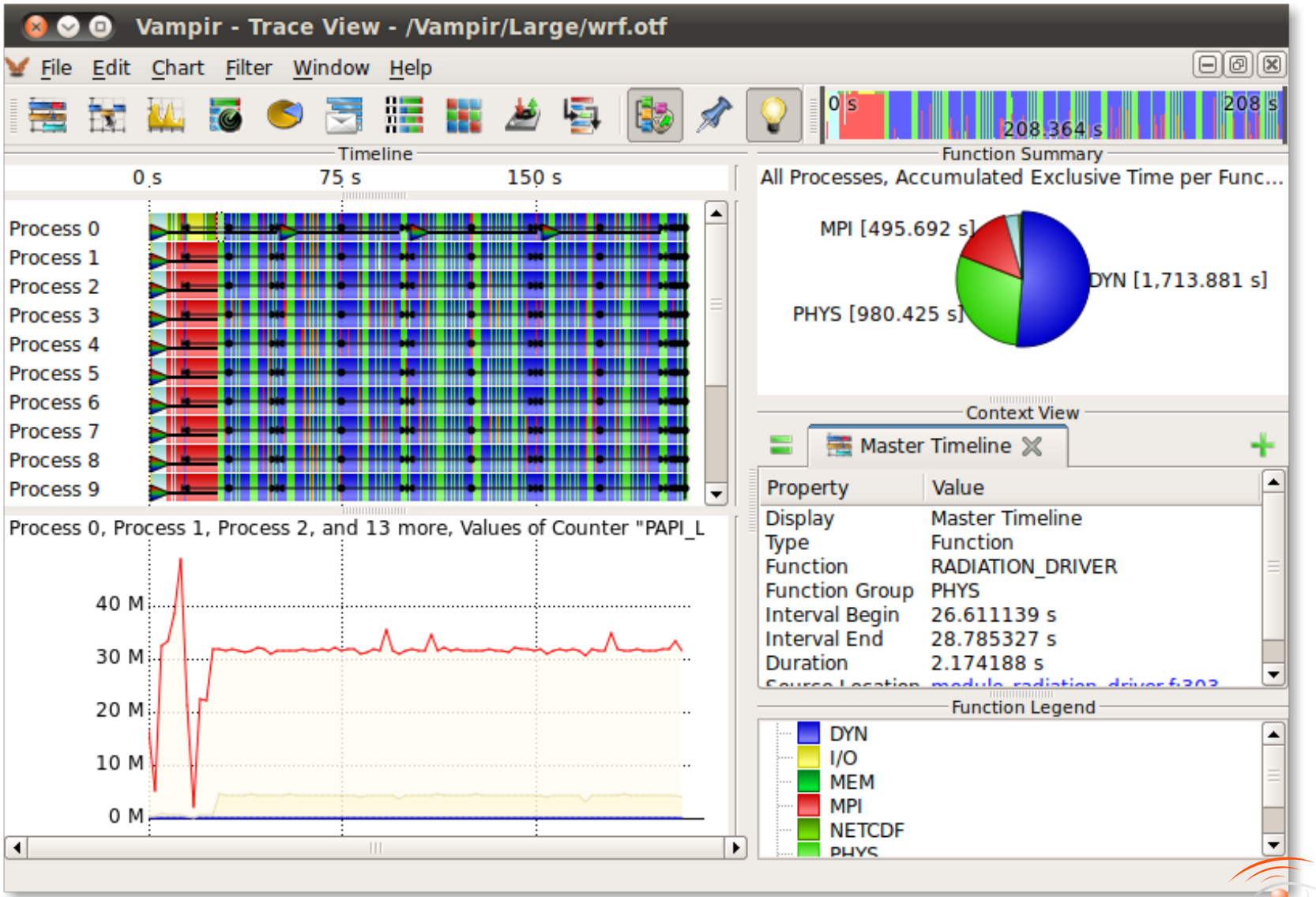
after execution you get an time ordered list of events

- we only provide function call tracing
- tracing though library wrapping

it is used for:

- runtime analysis of functions
- analysis of process concurrency
- visualization of program behavior
- ...

Visualization with Vampir



Pros & Cons

difficulties

- execution-time slowdown
- program perturbations
- amount of data volume

advantages

- very detailed
- summarized information can be computed for arbitrary time intervals
- useful for both performance tuning and debugging

Importance of good filtering

The GCC way

filtering in GCC

- *-finstrument-functions-exclude-file-list*=...
- *-finstrument-functions-exclude-function-list*=...

problems

- based on substring matching
- no wildcards
- no whitelisting
- filtering is imprecise

Test setup

- currently done during runtime
- proof of concept for new approach
- expensive instrumentation to guarantee compatibility

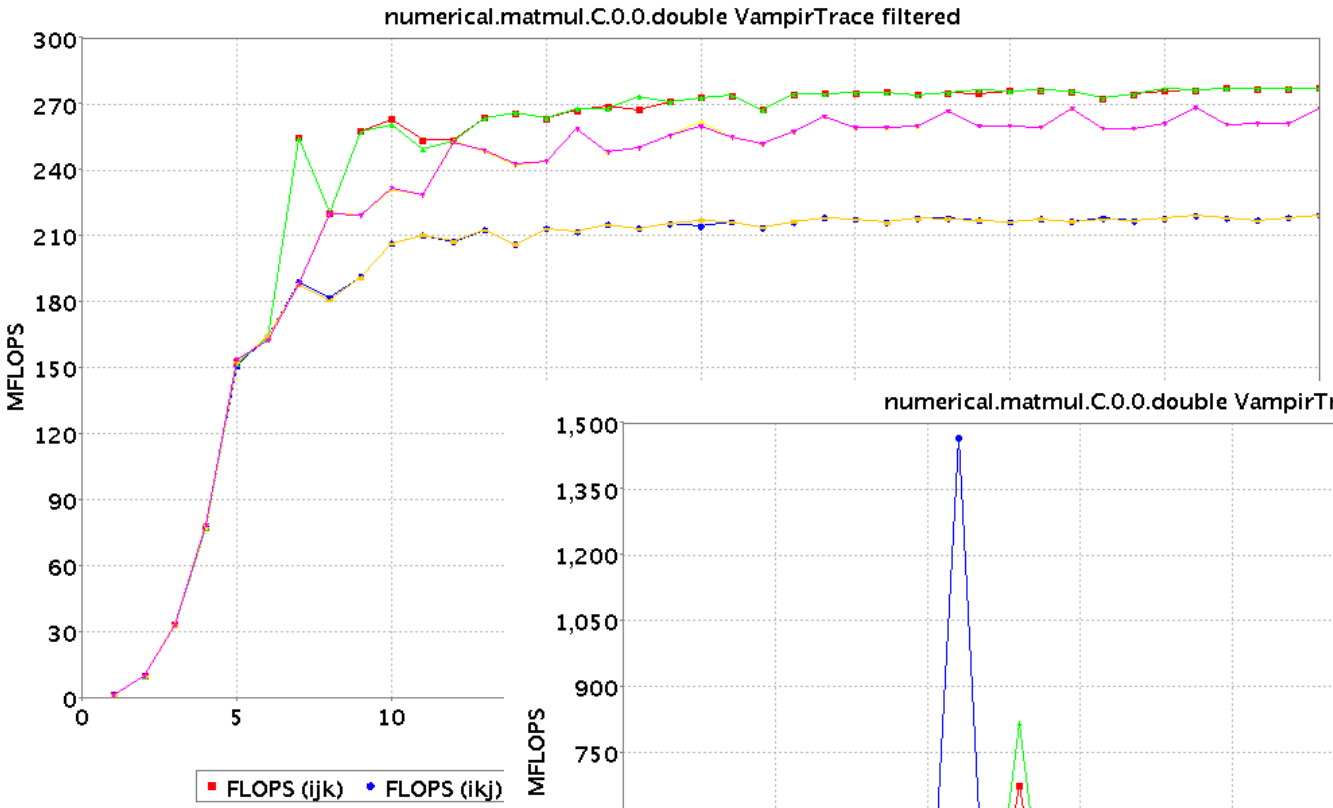
- SGI Altix 4700, Intel Itanium II Montecito 1.6 GHz
- measurement of matmul benchmark-kernel (single core)
- matrices of size 1x1 to 40x40, 150 iterations each

- use cases:
  - not instrumented
  - instrumentation via current approach filtered & unfiltered
  - instrumentation via InterAspect filtered & unfiltered

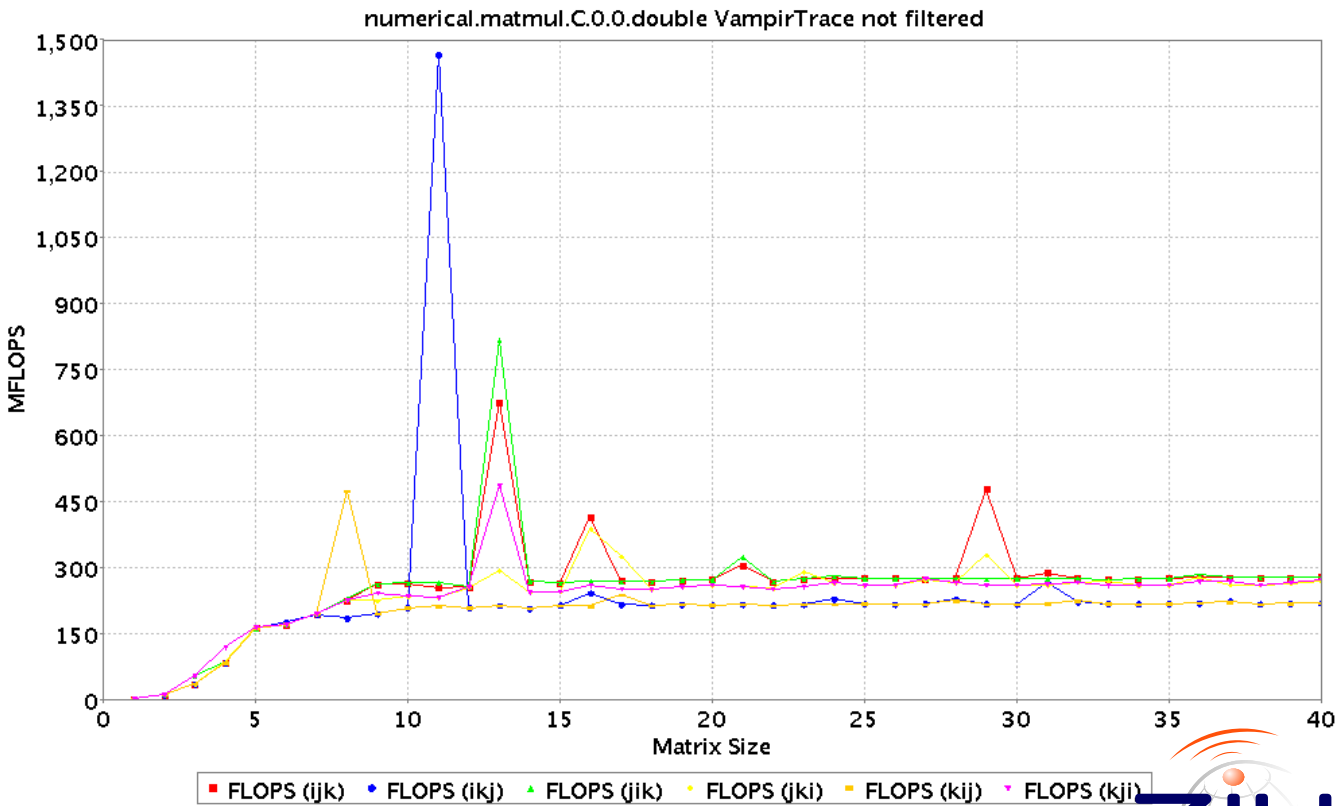
About InterAspect

- framework for code instrumentation
- developed by Stony Brook University NY
- current version: 1.0
- works with aspect-oriented programming
- creates GCC plugins
- licensed under the GNU GPL

Current monitoring system

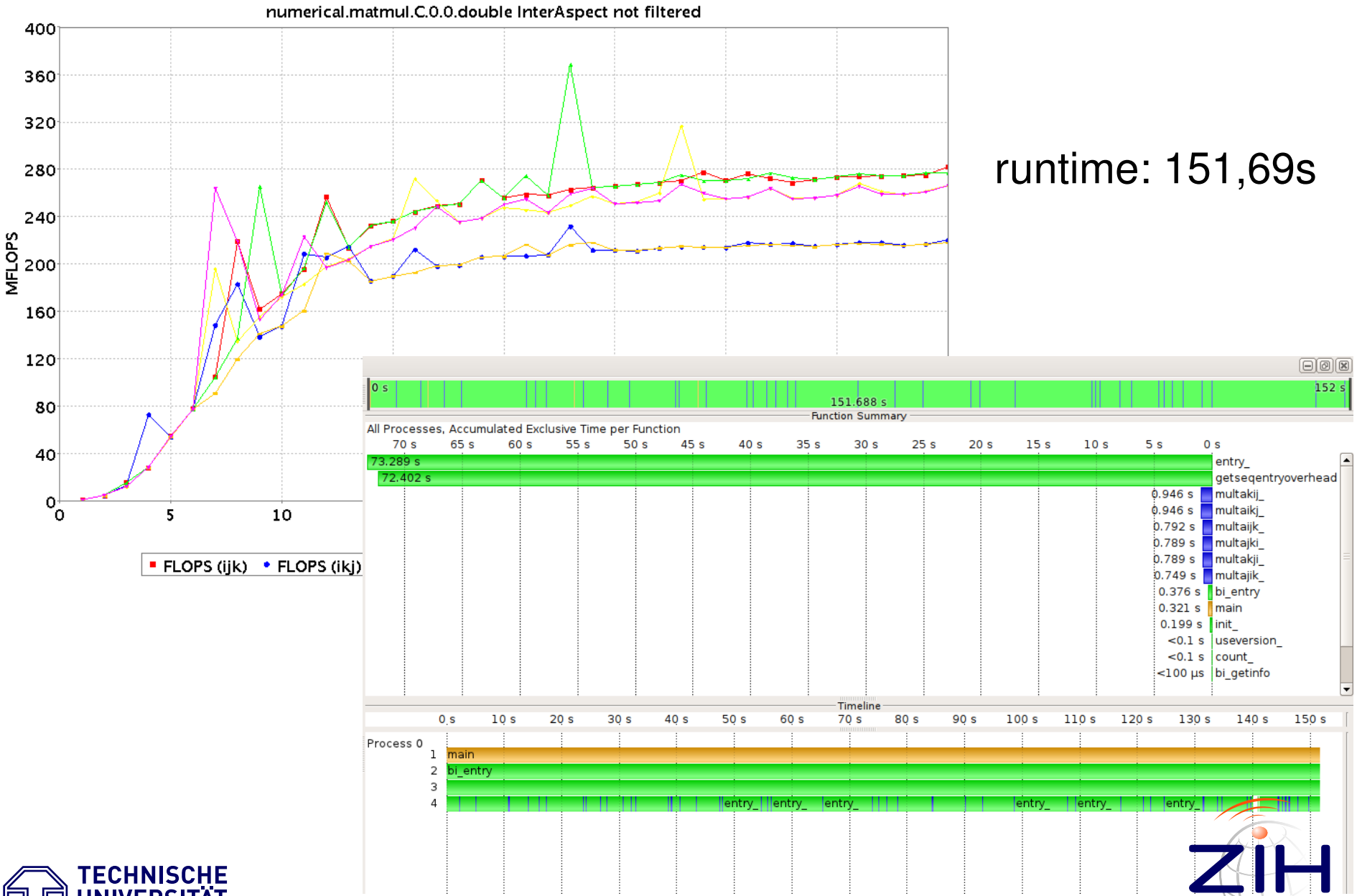


runtime: 38,45s



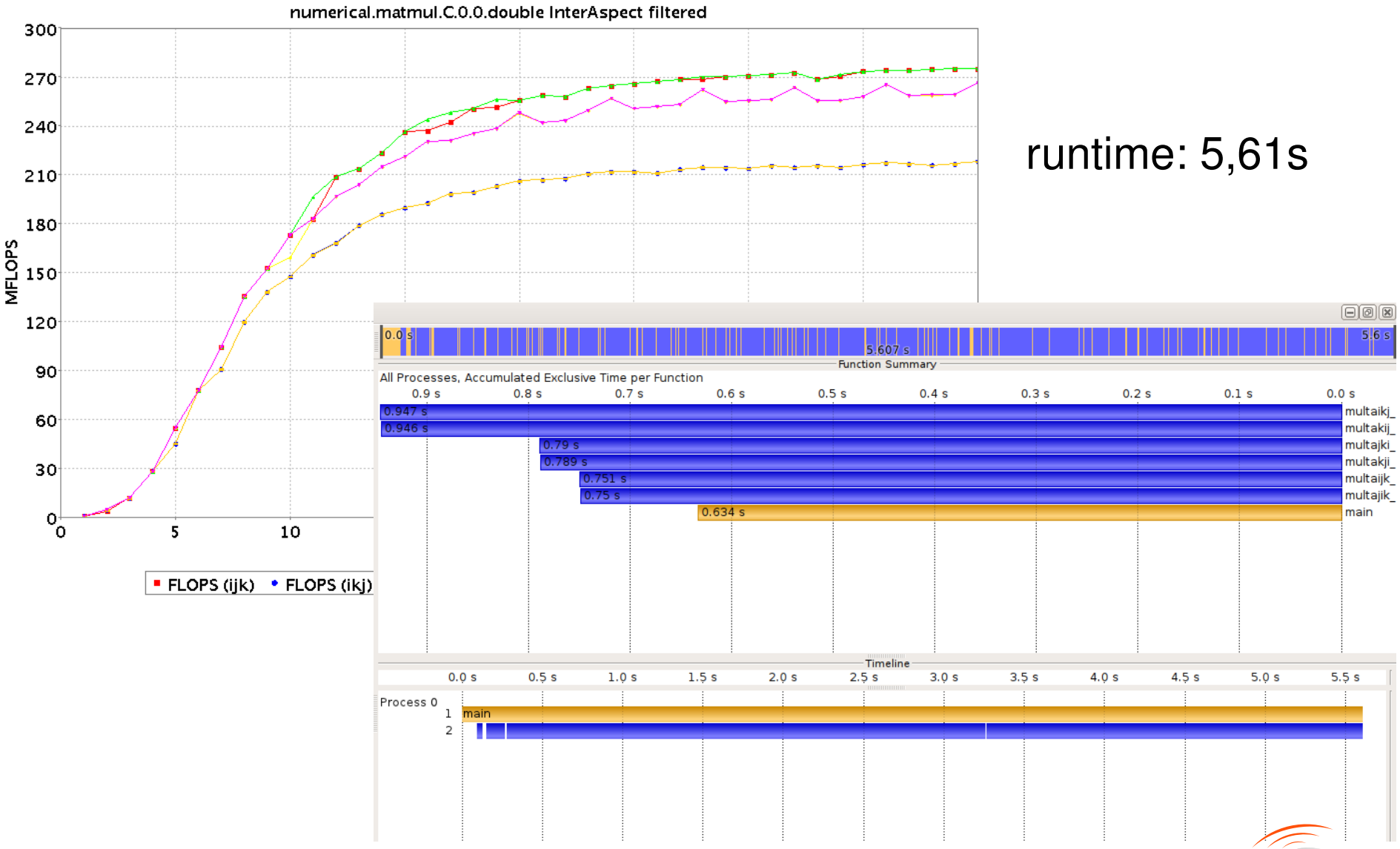
runtime: 43,32s

InterAspect unfiltered



runtime: 151,69s

InterAspect filtered



runtime: 5,61s

Importance of good filtering

- lesser data overhead generated
- only desired information are gathered (main, multa*)

- lesser result perturbations
- better reliability of results

- better runtime 5,61s vs. 151,69s (original 5,37s)

Our Vision

Why not using InterAspect anymore

- better linking between tracing environment and instrumentation
- no detour via GCC's instrumentation
- passing custom data to instrumentation functions
- more custom instrumentation of functions
- greater performance though lesser overhead

How GCC helps us

- Linux on 92% of Top500 HPC Systems
- GCC is default compiler for Linux

- an open source project
- one of few with open and extensible structure
- complicated but well documented internals

What we plan to do

- develop GCC instrumentation plugin
- joggle it with our Monitoring system
- compute function metedata during compilation

- provide filtering during compile time
- provide runtime filtering for debug purposes
- provide un-/instrumented function in binary
- switching at runtime
- (multiple optimization states based on power consumption in binary)?

Questions?