Large Scale Debugging

Project Meeting Report - May 2017

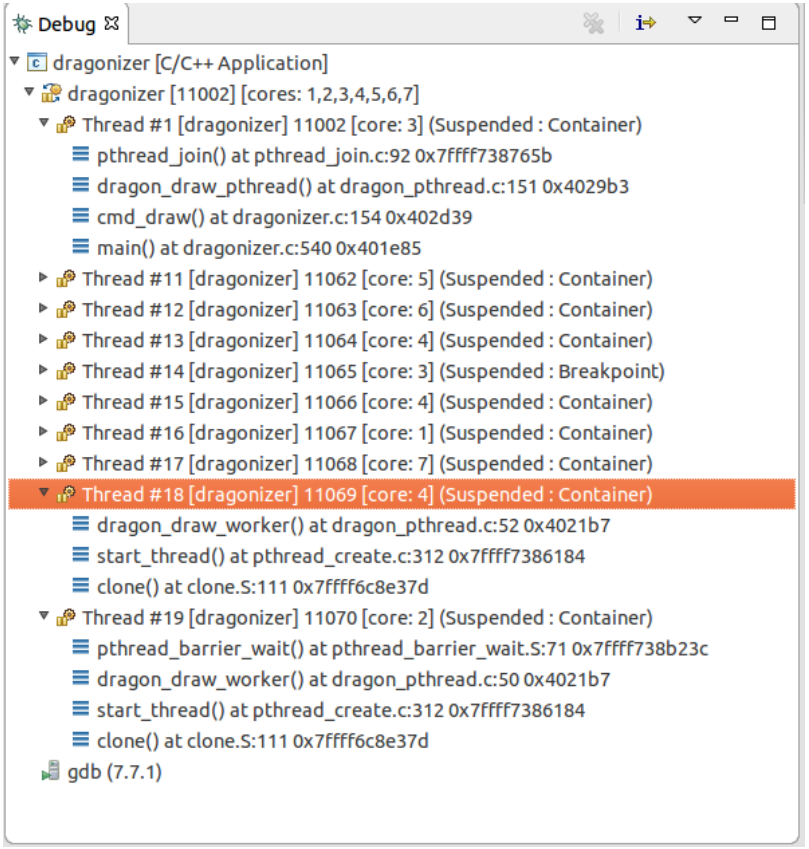
Didier Nadeau
Under the supervision of Michel Dagenais

Distributed Open Reliable Systems Analysis Lab
École Polytechnique de Montréal

Table of contents

Debug View

- The view shows the **program** being debugged, the **processes** running it and their **threads**.
- Threads shown in a **list**
- Must expand a thread to know its current **location**
- Similar to many different IDEs



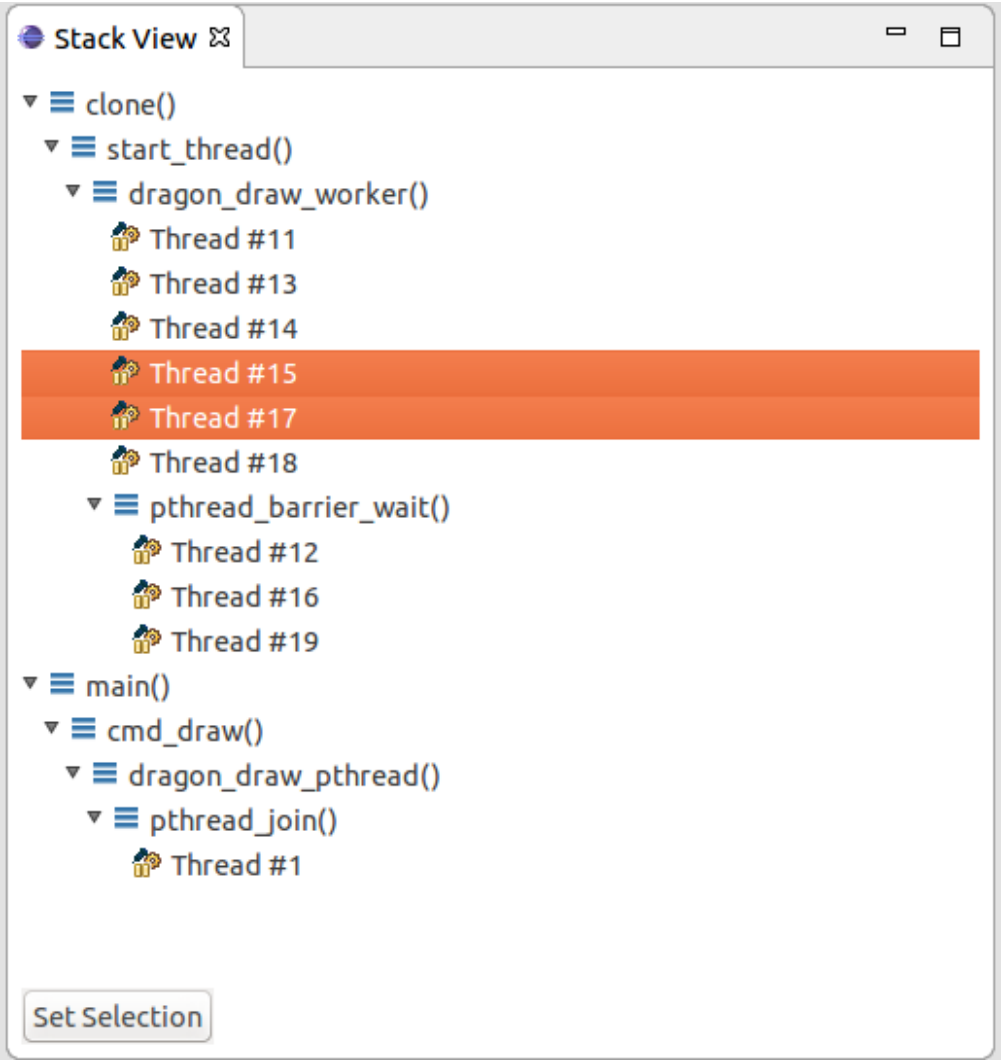
Stack Aggregation View

Automatic grouping

The stack aggregation view **automatically groups** threads using their call stacks.

- Logically groups thread together.
- Can be used together with the standard debug view.
- Lists threads and stack frames together.
- Can significantly reduce the number of elements in the view.

Stack Aggregation View



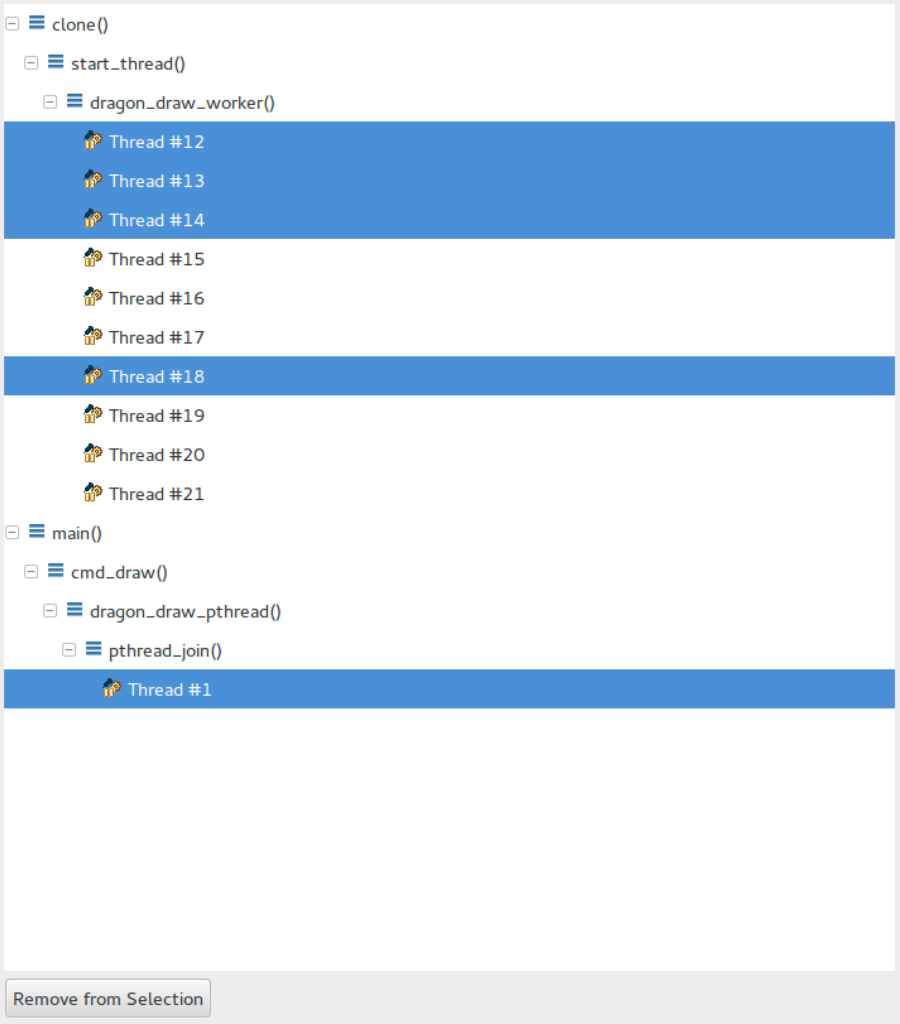
Dealing with large amount of threads

Many-core debugging

Large numbers of threads overload debug views and can generate an important number of events.

- User needs to scroll to view every thread.
- The developer can be overwhelmed by the amount of events
- Thread-specific or conditional breakpoints already filter out useless events.

Thread filter

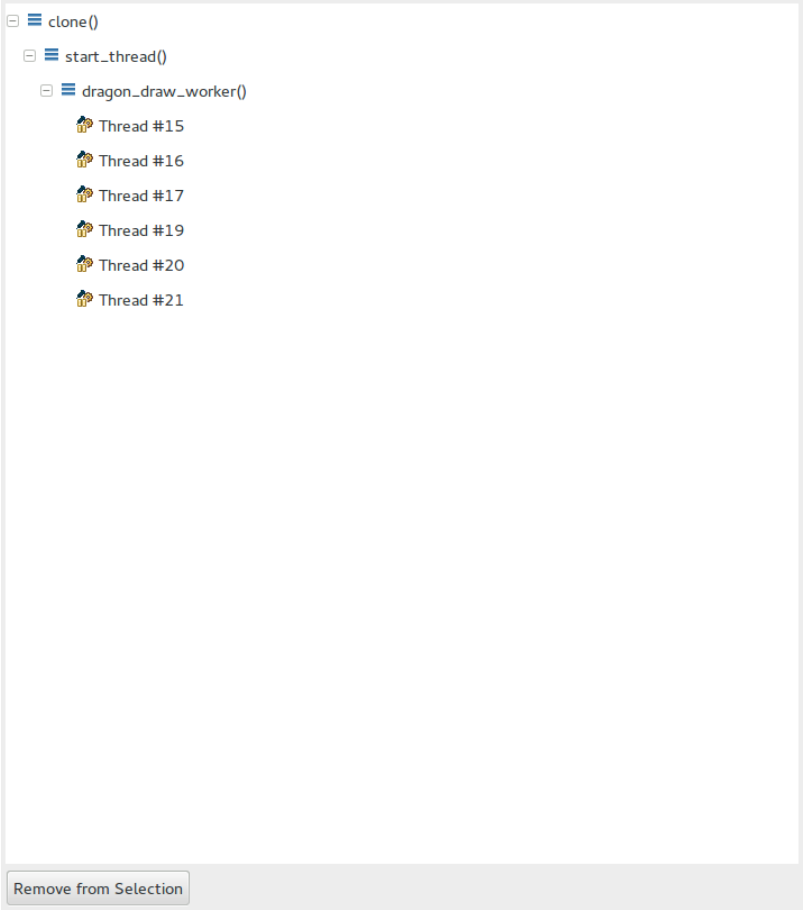


Selecting threads

- The stack aggregation view groups threads by their behavior.
- It can be used to select threads that do not interest the developer.
- For instance : Mask threads working in code that does not need to be debugged.

Thread filter

- The threads previously selected are **removed** from the view.
- The threads hidden from perspective **ignore breakpoints**, reducing event notifications.
- Affects **every view** of the IDE.



The filter removes unwanted threads from perspective.

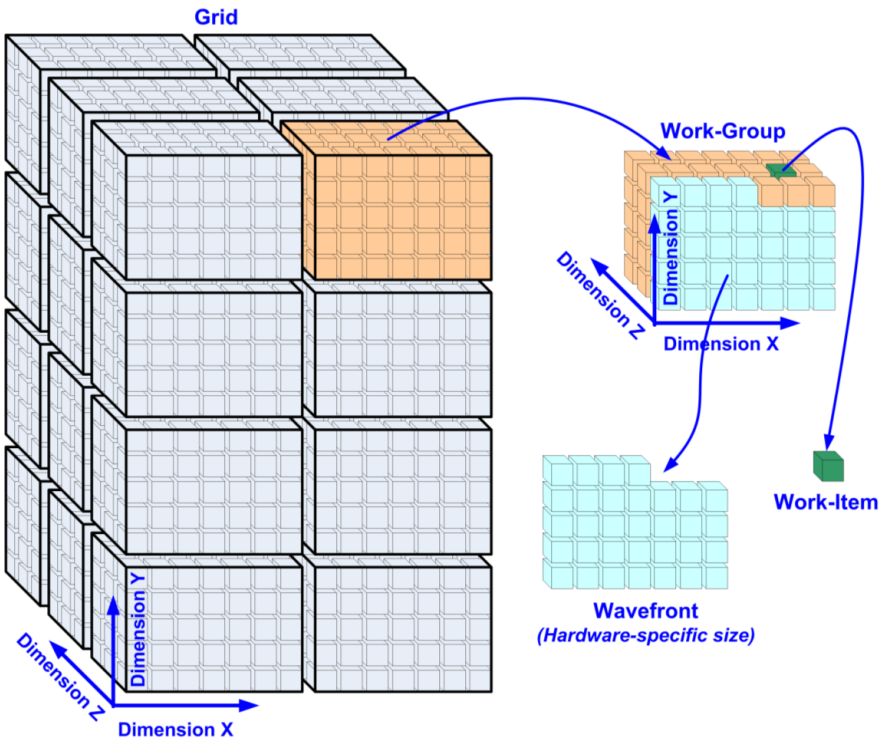
AMD Software Stack

Radeon Open Compute

An initiative launched in 2015 by AMD to provide an open-source software stack to interact with graphic cards for professional use and personnal use.

Heterogeneous System Architecture (HSA) Foundation

- Provides a standardized interface for programmer
- Multiple instruction sets
- Radeon Open Compute is an implementation by AMD



HSA Programming model [1]

- The programming model uses a **grid** filled with data elements.
- A function, or **kernel**, is applied to each element.
- Multiple elements processed together in a wave.
- The Radeon R9 Nano used can support more than **a thousand waves** simultaneously.

[1] From the HSA Programmer's Reference Manual 1.0

Displaying GPU waves

- The AMD R9 Nano has **4096** cores.
- Similar problem for displaying as threads, on a greater scale.
- There is no call stack in code running on graphic cards.

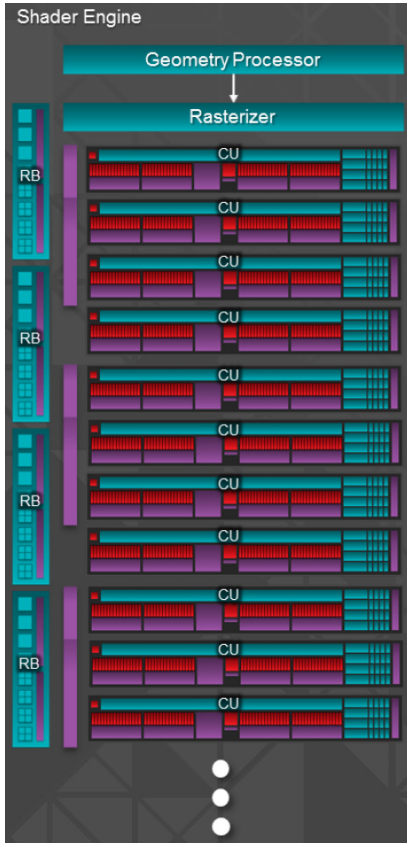


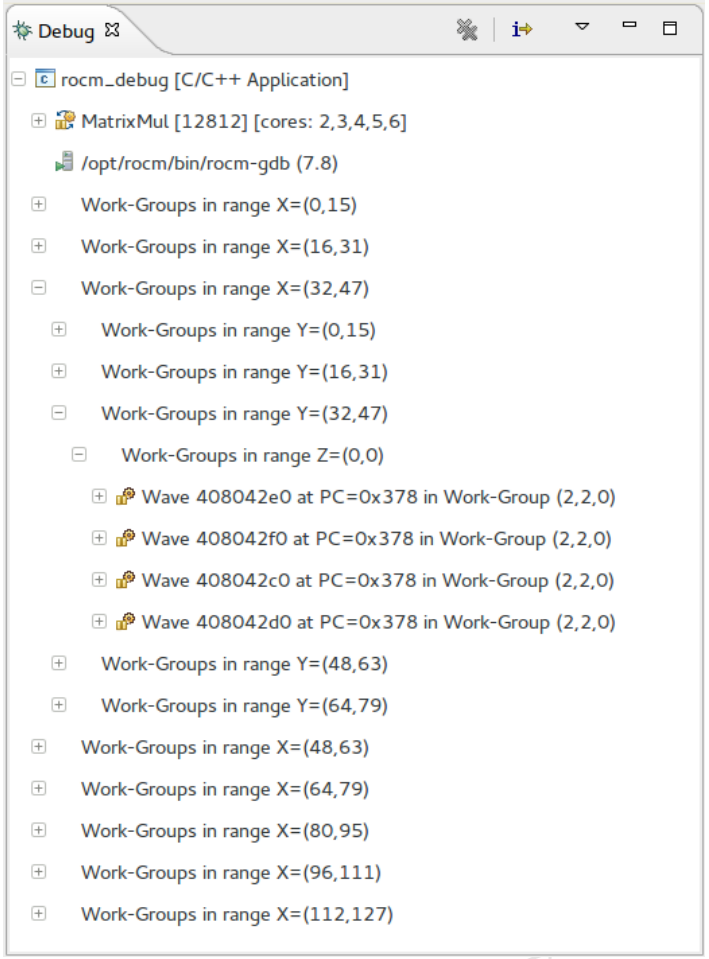
Diagram of Shader Engine [2]

[2] From anantech.com

Displaying GPU waves

Solution : take advantage of the programming model.

- The HSA programming model divides the waves in **work-groups**.
- It uses a **3 dimensions** grid.
- Similar programming model for **OpenCL** and **CUDA**.
- Group waves by their range along each axis.



Debug view with GPU waves

Wave filtering

Problem

Even with automatic grouping in the debug view, it can be tedious to inspect the waves currently executing.

Filtering

Problems in a kernel function can occur at specific ranges in the data grid. It can be interesting to limit the range that is in perspective to focus on these problems.

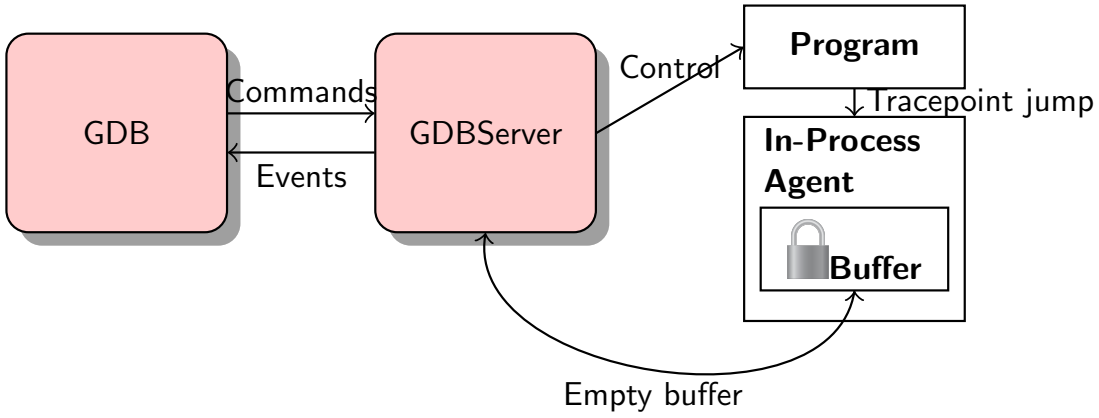
IPA Library

The In-Process Agent library is a shared object shipped with GDB. The goal is to move logic from GDB to a library loaded into the program memory space.

- GDB can be overloaded by **high-frequency events**.
- Traditional debugging involves costly **context switches**.
- GDB is single-threaded, can be overwhelmed by **multi-threaded** programs.

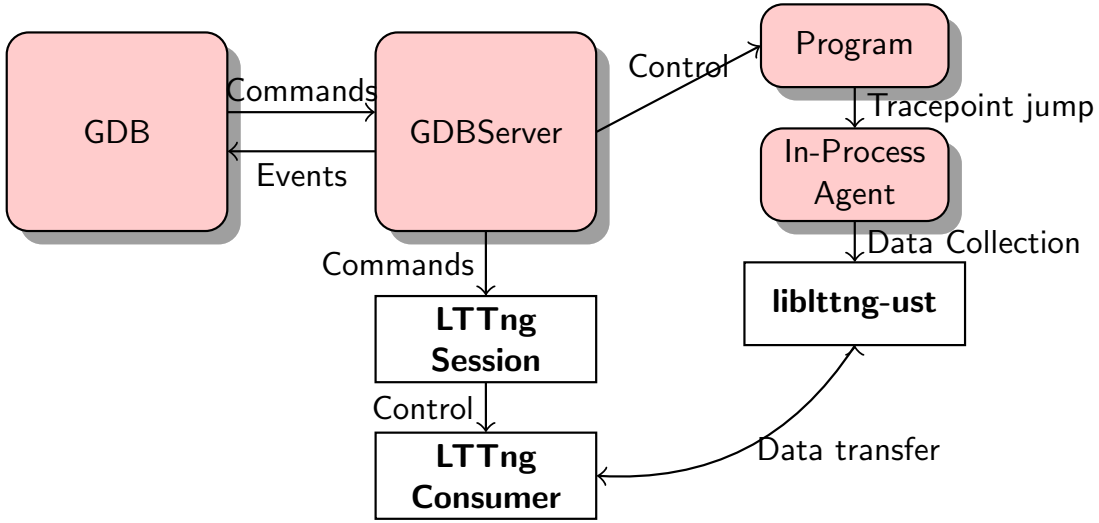
Fast tracepoints

- The IPA is used to enable **fast tracepoints**.
- Enable context-switch free collection.
- Scalability problems on multi-core processors.



Standard fast tracepoint architecture in GDB

LTTng dynamic tracepoints



Proposed fast tracing architecture using LTTng

- LTTng is used for tracing.
- Enables lock-free collection.
- Circumvents static event definition by modifying the trace header.

Conditional breakpoints

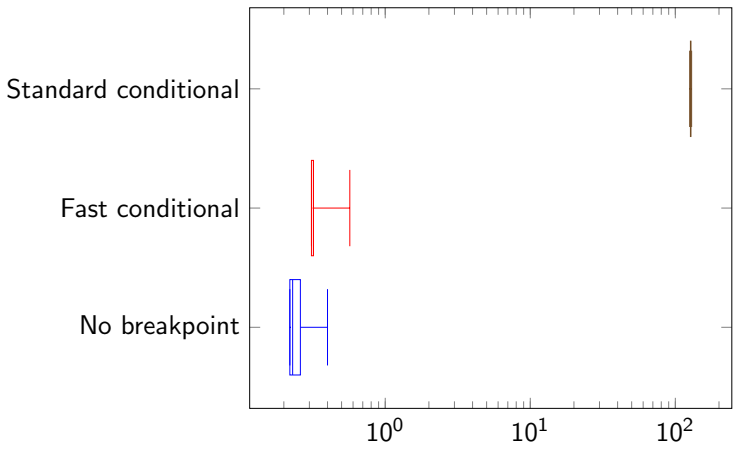
Behavior

When a conditional breakpoint is hit, the program stops and GDB is called.

Cost

The cost is very important as it involves context switches. GDB can quickly become a chokepoint if the condition is frequently evaluated.

Fast conditional breakpoints



Total execution time in seconds for a program where a breakpoint condition is evaluated a million time in a loop

- A prototype that uses the IPA to evaluate conditions for breakpoints has been tested.
- The test program does calculation in a loop where a conditional breakpoint is inserted.
- The overhead is extremely important for standard conditional breakpoints.

Any Questions ?

Contact

didier.nadeau@polymtl.ca