Targeting and Tuning Distributed CnC

James Brodman
james.brodman@intel.com

Outline

- What is Distributed CnC?
- Targeting Distributed CnC
  - Serialization
  - Handling “Exotic” Data – Pointers, Arrays, Etc.
- Tuning Distributed CnC
  - Execution
  - Communication
- Example
- Tools

What is Distributed CnC?

- Distributed CnC (DistCnC) is an extension of Intel's Concurrent Collections for C++ library that targets distributed-memory machines
- Any DistCnC program is also a shared-memory CnC program
  - Steps, Tags, Items
  - Puts, Gets
  - The Language is the same

DistCnC Execution

- Heterogeneous parallel execution model
  - More like MPI+OpenMP than MPI
  - Each address space has a full shared-memory runtime to take advantage of multicore
    - Powered by TBB
- Common communicator interface allows multiple backends:
  - Sockets
  - MPI

Targeting DistCnC

- Porting CnC programs to DistCnC only really needs one thing:
  - Serialization routines for user data types
    - What bits in objects need to be communicated between address spaces
    - Takes a CnC::Serializer object
  - Routines for primitives and common utility DTs built into the runtime (std::pair, std::vector, int, double, etc)

Serializing Arrays, Pointers, and the Kitchen Sink

- DistCnC provides convenience mechanisms for serializing objects, arrays, and pointers.
- Macros
  - CNC_BITWISE_SERIALIZABLE(T)
  - CNC_POINTER_SERIALIZABLE(T)
- Arrays
  - DistCnC provides a Chunk object for handling arrays
  - Base Type must be serializable
  - The programmer can choose to let CnC handle allocating memory on remote processes
- Global Read-only Data
  - Can be distributed by adding a serialization method to the graph context

Tuning DistCnC

- DistCnC allow exposes Tuner objects that provide programmers with mechanisms to control the execution and communication of a program

- Execution -> Step Tuners
- Communication -> Item Tuners

Step Tuners

- Step tuners let programmers control:
  - When a step should execute
  - Where a step should execute
- When (depends):
  - A step should not execute until the programmer specified data items are available.
  - This is not unique to DistCnC, but is particularly useful since checking for data can be more expensive in DistCnC
- Where (pass_on):
  - By default, CnC distributes step instances to processes round-robin.
  - Programmers can override this and specify how to distribute steps

Item Tuners

- By default in DistCnC, data is pulled from remote processes when a step executes in a particular location.
  - Broadcast to ask who has the data, then receive it.
- Item tuners let programmers specify the consumers of a data item.
  - Consumed_on
- Allows runtime to send data to the processes that will execute those steps.
  - Push instead of Pull
  - Probably going to be used with Step Tuner(s)

Example - MMM

```
<ijk> -> (Do_mults)
[a], [b] -> (Do_mults) -> [t]
<ij> -> (Reduce_tiles)
[t] -> (Reduce_tiles) -> [c]
Env -> <ij>, <ijk>, [a], [b]
```

- Do_mults – performs matrix-matrix multiplication of tiles A_{ik} and B_{kj} to produce T_{ijk}
- Reduce_tiles - Adds tiles along the k dimension to produce tile C_{ij}
- Item collections will have data with type Tile
  - Tile contains a statically sized 2D array of doubles

Targeting MMM to DistCnC

- Make sure all data is serializable

```
struct Tile {
      double data[TILESIZE][TILESIZE];
};
```

  - Tile – only member data is a 2D array of known size.
    - CNC_BITWISE_SERIALIZABLE(Tile);
    - Programmer could also write a serialize method.
    - Want to use Tile * instead of Tile?
      - +CNC_POINTER_SERIALIZABLE(Tile);

- Great! Program can now execute using DistCnC.

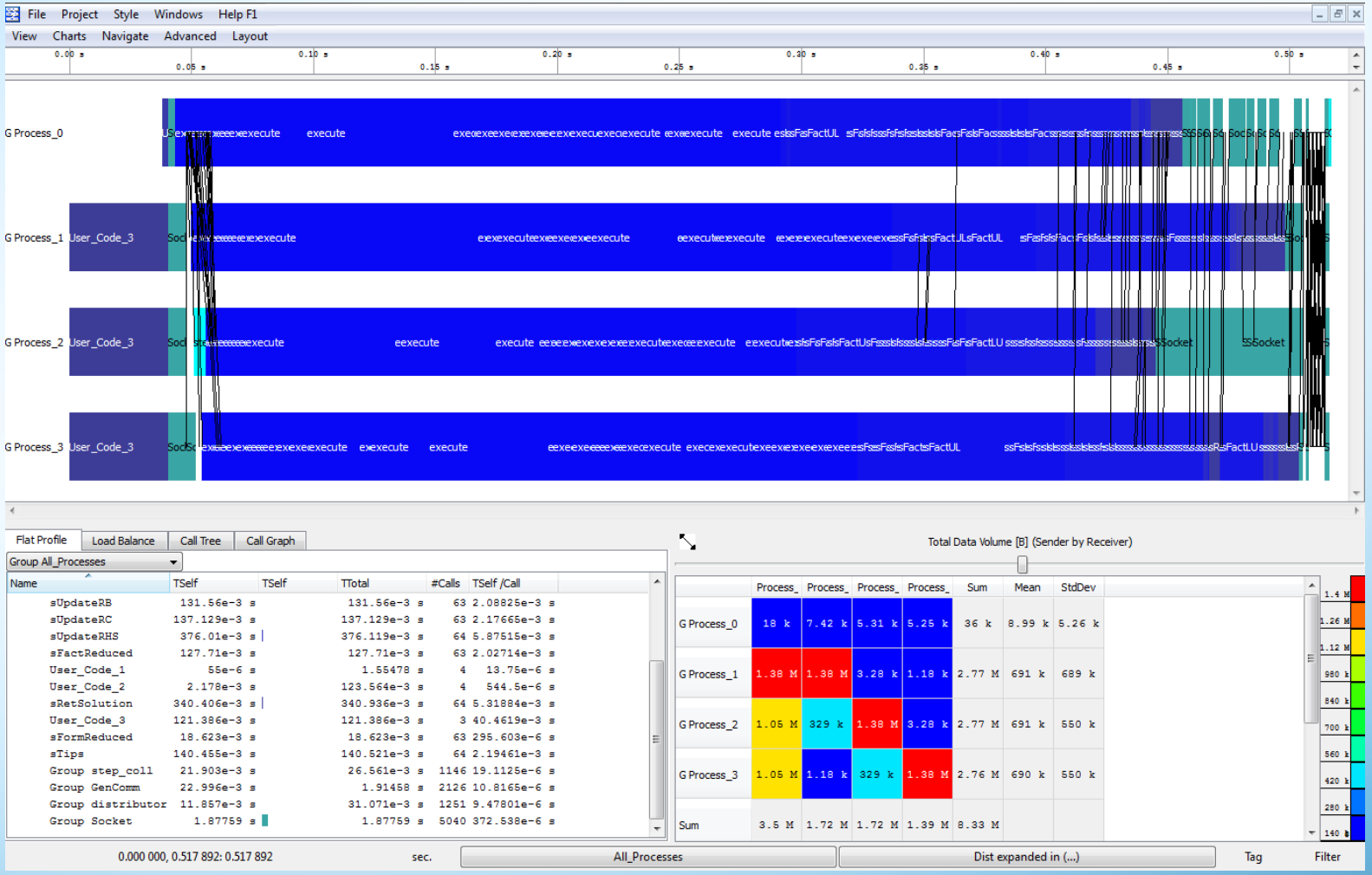
Tuning MMM on DistCnC

1. When
   - Step Tuner for collection (Reduce_tiles)
   - Don't let (Reduce_tiles) execute to compute C_{ij} until T_{ij0}… T_{ijn} are available.
2. Where – the programmer wants to minimize communication, so compute C_{ij} and T_{ij0}…T_{ijn} in the same place.
   - Step Tuner for collections (Reduce_tiles) and (Do_mults)
3. Communication
   - The programmer knows which step instances will consume A_{ij} and B_{ij}
   - Item Tuner for collections [a] and [b]
   - Avoids broadcasts from pulling data

Tools

- DistCnC has support for generating execution traces of programs using Intel's Trace Analyzer and Collector
  - http://software.intel.com/en-us/articles/intel-trace-analyzer/
- These traces show:
  - Which steps run where and when
  - What communication happens when, between which processes does it happen, and how much data is sent
- Works with both Sockets and MPI
  - Also useful for shared-memory!
- Built-in to Intel CnC 0.6 on Whatif

Intel Trace Analyzer and Collector



Conclusion

- DistCnC lets programmers bring their CnC programs to larger machines.
- Very low initial programmer cost
  - Serialize and Ready-to-Go
- Provides several mechanisms for tuning program performance
  - Execution
  - Communication
- Open Questions
  - What types of CnC programs profit from distribution and which do not?
  - What else is useful or necessary to get high performance programs that work on both shmem and distmem?

Questions?