Assignment #4: Data Collection and Preparation

Caitlin Ross & Noah Wolfe
02/18/2016

1 Data Source (ROSS)

In this assignment, we have decided to generate/collect data from our ROSS research projects. ROSS is a massively parallel discrete-event simulator that can process billions of events per second [2], [1]. ROSS models are made up of a collection of logical processes (LPs). Each LP models a distinct component of the system. LPs interact with one another through events in the form of time-stamped messages. An MPI task is abstracted as a processor element (PE) in ROSS. Each PE owns a number of LPs and schedules events in time-stamp order for all LPs assigned to it. Events that are destined for a logical process on another PE (i.e. remote events) are sent as MPI messages.

The underlying ROSS simulator is itself, a very complex piece of software with a large set of parameters. When changed, these parameters can trigger large changes in metrics such as number of reverse computations, remote events, event efficiency, etc. Currently, the only data collection is averaged over all processes and collected at the very end of the simulation.

2 Research Questions

ROSS is designed to be a very fast and efficient simulator. In order to get the best possible speed-up, developers need to know what is going on behind the curtains throughout the simulation. We need to know not only if a simulation is experiencing hotspots, but also be able to detect the cause of those hotspots. This requires collection of data throughout the simulation. Two research questions that can be solved by analyzing ROSS runtime system data are:

- Where is my simulation spending it's compute time?
- Which ROSS scheduler parameters are the most influential to the minimization of remote events for a given model?

The first research question should be answerable by simply visualizing the large set of collected data. The second item will require much more data manipulation and analysis such as a weighting system to find correlations between ROSS parameters and simulation speed.

3 Hypotheses

The results to the above research questions are very dependent on the given model so we will focus on using the Slim Fly network model which simulates a large cluster of compute nodes connected and communicating in a Slim Fly topology. Simulating a uniform random workload and using the minimal routing algorithm, we expect to see the majority of the simulation compute time being spent performing forward events. This hypothesis comes from prior knowledge of the mapping of LPs to PEs for the Slim Fly model in ROSS. A uniform distribution of compute nodes, routers, and MPI processes means work

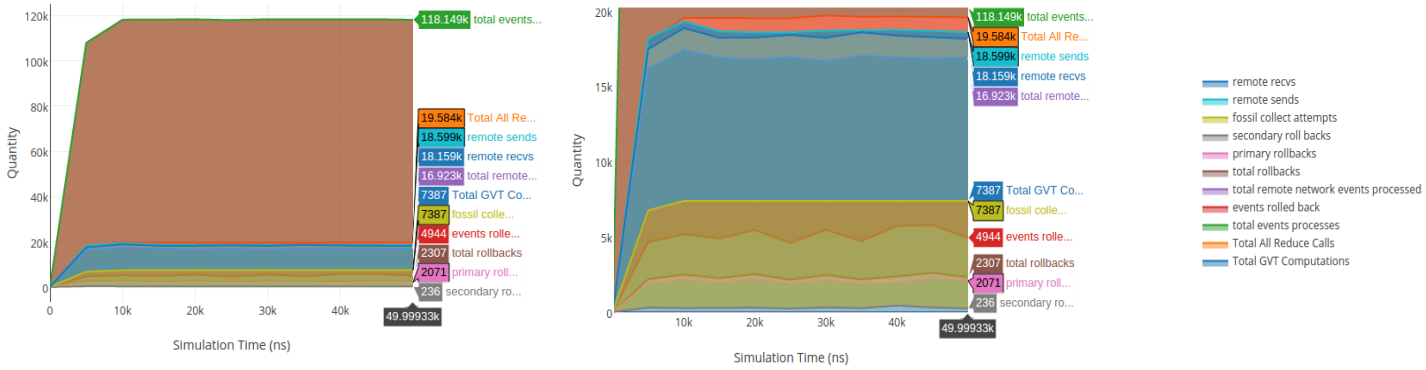
should be evenly distributed and result in very little rollback activity. In the case of the second research question, we expect to find that the batch size parameter has the most influence on limiting remote events. This prediction is based on prior experience playing with the batch size to minimize execution time.

4 Data Format, Extraction & Manipulation

Using the simple csv format, the raw data is structured so that the columns represent the various ROSS parameters and metrics and the rows represent those same parameters collected at a later instance of simulation time. The default sampling frequency is $0.01 * simulation\_end\_time$ but can be adjusted with a commandline flag "–report-interval=n" where $n \in \{0.999, ..., 1/simulation\_end\_time\}$. Each MPI process creates it's own data file and appends the MPI rank number to the end of the filename. Filenames can be adjusted with the commandline flag "–stats-filename=string". We are currently collecting a total of 20 ROSS simulation metrics (20 columns) and can sample up to 50,000 points in time for 4 MPI processes. This configuration results in 4 million data points. Even larger data sets can be generated with longer simulation runtimes.

5 Sample Visualizations

The first visualization shown in Figure 1, was created using the Plotly web-based visualization tool and is used to answer the first research question posed in section 2. The figure displays the total events computed in the simulation as well as the number of events computed for specific event types such as rollbacks, fossil collection, etc. As can be seen in the figure, the number of All Reduce computations exceeds all other types of computation so it appears the majority of the time is spent performing MPI_All_Reduce.



(a) Overall View

(b) Dense Section Zoomed View

Figure 1: Visualizations of the ROSS data set showing the number of events computed in each section of the simulation over the length of the simulation. Sub-figure 1a shows a global view of all metrics while sub-figure 1b shows a view zoomed in on the dense region at the bottom of sub-figure 1a.

6 Visualization Tool Review

References

[1] D. W. Bauer Jr., C. D. Carothers, and A. Holder. Scalable time warp on blue gene supercomputers. In *Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, PADS '09, pages 35–44, Washington, DC, USA, 2009. IEEE Computer Society.

[2] A. O. Holder and C. D. Carothers. Analysis of time warp on a 32,768 processor ibm blue gene/l supercomputer. In *in 2008 Proceedings European Modeling and Simulation Symposium (EMSS*, 2008.