Research Report

Monday 03/31/2014

Wei Wang

What I needed to do

- Looking at Community Detection Code in more detail by experimenting with the newly received large input file
- Looking into Rose compiler whole application auto-tuning framework

Progress & Problems

1. The 2 minute program (with the large input file) takes 45 seconds (37.5%) to do Input processing. The compute intensive part takes about 70 seconds (58.3%) and is mostly spent in Phase 1 of all 6 phases.
2. The following shows the energy info of the same function called in phase 1 to 6 (highlighted in white).

```
Loop 0 <line-97> - Time 45.985516 Total energy consumed 2567.676585 Ave. Power Level 55.836638
Loop 1 <line-100163> - Time 66.480876 Total energy consumed 5088.658095 Ave. Power Level 76.543186
Loop 2 <line-200163> - Time 2.555154 Total energy consumed 241.563885 Ave. Power Level 94.539836
Loop 3 <line-300163> - Time 0.763129 Total energy consumed 92.764410 Ave. Power Level 121.557918
Loop 4 <line-400163> - Time 0.322795 Total energy consumed 42.374580 Ave. Power Level 131.274132
Loop 5 <line-500163> - Time 0.097034 Total energy consumed 12.649350 Ave. Power Level 130.360405
Loop 6 <line-600163> - Time 0.052281 Total energy consumed 6.791205 Ave. Power Level 129.898236
Application(EnergyStat) - Time 122.123425 Total energy consumed 8492.240460 Ave. Power Level 69.538178
```

3. Choosing different code versions to execute won't be much better than choosing a version that works best for the first (common case) phase. But the first phase involves many smaller sub-phases, which needs to be studied

Issues

1. The compute-intensive part seems to spend a significant portion of time in maintain map structure.
2. The code involves indirect and even double indirect access of array elements.
3. Still haven't started to test ROSE autopar and autotuning framework yet

The plan

- Looking into Phase 1 of the code in detail. Figure out ways to (auto)tune the code
- Looking into Rose compiler whole application auto-tuning framework
- Get a paper out!