Weekly Research Report

Monday 10/21/2013

Wei Wang

review of the issues to solve/What I needed to do

- Although Lulesh can go through polyhedral compilers, no extensive experiments were done, no results (numbers /graphs) were generated
- IMPACT 2014 workshop on polyhedral compilation techniques due on 10/25/2013 AoE, need to 1) get lulesh results into paper & 2) get a ready draft to John by Monday.
- Manual implementation of cardiac code (for Plos ONE submission) is not well vectorized

progress on: IMPACT 2014 draft

- Added MIC results of Cardiac code to the draft
- Redo all parts of the draft except Related Work and Conclusion
- Sent the draft to John
- Need to work on related work and conclusion
- Depend on how many pages we want, we can add some discussion of lulesh challenges

progress on: Lulesh

- Generated program variants for one SCoP
  - that scop occupied ~12% execution in fully parallelized OpenMP
  - did not generate variants for more SCoP because the polyhedral compiler took too long to do transformation
  - The results on elo is that the variants performed as well as the original OpenMP implementation on Sandy Bridge
  - The original OpenMP program had slow down running on MIC.
  - Not planning to add lulesh results to IMPACT2014 draft

progress on: PlosONE

- Met with Will on profiling the cardiac code using Intel Vtune Amplifier
  - we found that the vectorization is at the OK range.
  - The problem is related to memory stalls and also the bandwidth is not 100% utilized.

The plan (before Thursday)

- Related work and conclusion section of IMPACT 2014 workshop paper
- Try to get good Lulesh numbers using polyhedral approach still
  - at least I will have the energy and time correlation data generated on Sandy Bridge (on MIC, the problem is aforementioned: the program is not performance portable yet: needs investigation)

Weekly Research Report

Thursday 10/17/2013

Wei Wang

What I needed to do (review)

- work on using PPCG compiler to transform Lulesh programs
  - worked on it (details in next slides)
- work on the vectorization of the cardiac code
  - Did not work on it yet.

What did I do (1)

- work on using PPCG compiler to transform Lulesh programs
  - Riyadh's PPCG could auto-parallelize (involving scalar privatization) the sequential code while tiling it. But the problem is: the time it took to auto-parallelize is at the magnitude of hours!
  - Solution: ?

What did I do (2)

- Polyhedral optimize cardiac code for input size smaller than 2048, on Xeon Phi
  - Previously, input size 2048 had barely speedups (about 3-5% improvement)
  - Found that for smaller input size, the improvement was about 15%. For example, size 1024, the 110X speedup got improved to 110*(1+15%) = 130X speedup by using polyhedral transformations. (Graphs attached in the beginning)

What do I plan to do?

- Get lulesh auto-parallelized and transformed for MIC, put results into paper draft
- Get a ready to submit version to you by Sunday (10/20/2013) for IMPACT 2014 workshop on polyhedral compilation techniques (deadline 10/25/2013 AoE)
- Get some progress on improving manual implementation of cardiac code.
- Start looking at Windows PoCC/PPCG