Suggestions (Dan):

* Omkar, Naveen - Let’s move to markdown or some other text-based so that we can more nicely see what gets changed between revisions. Talk w/ Andrew if you need any suggestions.
* Naveen: please brief the rest of us on the due dates for the SC15 conference (Texas-Austin). This is the conference where we want to submit.
* Put some dates on the timeline so that we have some rough plan in place.
* Let’s make each task become a short technical report. We should avoid a situation where we wait till the end to summarize our findings since we forget in April what compile flags we used in January.

**12/07**

Dan Negrut, Hammad Mazhar, Arman Pazouki, Ang Li, Omkar Deshmukh, Naveen Anand

Relevant Problems:

* Sparse Matrix-Vector Multiplication
* LSS – Ask Arman about the bottleneck
* Sparse LU factorization – Ask Ang about the bottleneck
* Neighbor Search – Ask Arman about the bottleneck

Start with easier, primitive problems like sorting

Order in which we are going to benchmark:

1. Benchmark Memory bandwidth performance
2. Sort, Scan, Reduce
3. Linear Algebra – Vector-Vector multiplication, Sparse Matrix-Vector multiplication
4. Sparse LU factorization – F/W B/W substitution
5. Neighbor Search

Keep code local until we publish results

Both code and documents go on Github

Make a directory in the repository for minutes and documents – Add progress log to documents

Link to SBEL project page

Next meeting on Friday - 12/12

**12/05**

Description of platforms in terms of underlying H/W and execution environment supported.

|  |  |  |
| --- | --- | --- |
| **Platform** | **H/W** | **Environment** |
| AMD CPU | AMD Opteron 6274 | OpenMP, OpenCL |
| Intel CPU | Xeon E5-2690v2 | OpenMP, OpenCL |
| MIC | Xeon Phi | OpenMP, OpenCL |
| Tesla GPU | Tesla K20x | CUDA, OpenCL |
| GTX GPU | GTX 770 | CUDA, OpenCL |
| Jetson | Kepler GK20A | CUDA, OpenCL |
| APU | AMD A10-7850K | OpenCL |

We will test for following set of problems, in terms of absolute FLOPs rate & memory bandwidth and  utilization w.r.t theoretical peak.

* Matrix - Matrix multiplication
* **Sparse Matrix - vector multiplication (2)**
* **Sparse LU factorization (3)**
* **F/W B/W substitution (3)**
* Sorting and reduction (Low priority)
* **Neighbor search**
* **Sort, scan, reduce (1)**
* **Memory B/W (0)**

This gives, for each problem, 3 implementations - one each for OpenMP, OpenCL and CUDA. Depending upon platform, they may further need to be modified and used with respective tools and compiler settings. This results in 13 unique tests per problem (6 Native + 7 OpenCL), with 65 tests in total.

TODO:

1. Decide whether these problems are relevant and if we need to add/remove anything to the list.
2. While testing, we need to be mindful to only test with inputs that are important to us. Hammad, Ang and Arman to provide input matrices and test data.
3. It is not clear to me how to make **SPH** part of this. From what I understand, the current bottleneck is not something we can build from smaller blocks that we can test across different platforms.

**Repository structure:**

**Top level :**

1. Problems
2. Cumulative Results
3. Progress log in SVN

**Problems :**

* Will have separate folders for each problem/algorithm we are working on.
  + Every problem folder will have folders for different H/W environment.
    - Every environment folder will have
      * Code - On Github
      * Inputs - On Box
      * Results (Text file or a table)
      * README (Helps setting up the platform and running the code)

**Cumulative Results :**

* Contains a table and graph comparing the results we have obtained.
* Updated every time we obtain results from a new problem or hardware.

**Other ideas:**

* We can convert the git repo to a web page
* **SBEL project page**
* Who all should have access to the repo? Private
* Should we have a comments section for the users?
* Should we provide contact information? If so, who should the users contact?

**12/12**

Took one problem : reduce– float array 1,000,000 values – got results in some platforms

OpenMP is home-brewed code. Tried 3rd party library code on the rest of the platforms.

Takeaways:

OpenMP is faster – Thrust version is faster than our code implementation

Run code for float and double

Long and long long would be useful to us

Xeon Phi is particularly bad – unexpected

Should we come up with a normalized metric?

For every platform, we will specify the metrics ( and compiler ) – Memory, clock, flop rate etc.

Think about organize the results so we mine it. – But don’t do it now

Take outputs for different sizes – up to 25M

Generate a project webpage (like a blog) and populate results into it every once in a while – nice plots, clear, bulleted, short