LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-SR-702813

LLNL Center of Excellence Work Items for Q9-Q10 period

J. R. Neely

September 13, 2016

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL Center of Excellence Work Items for Q9-Q10 period September 1, 2016 – Feb 28, 2017

1 Milestone COE5: Target description from SOW

"*Half yearly activities report, next period execution plan defined.*"

This work plan encompasses a slice of effort going on within the ASC program, and for projects utilizing COE vendor resources, describes work that will be performed by both LLNL staff and COE vendor staff collaboratively.

1.1 Overview and Update

As of the end of Q8, most ASC code projects/teams have efforts underway for Sierra preparedness, in many cases with support of the COE vendors. Knowledge transfer between those working on the unclassified applications to those targeting classified applications is an ongoing activity within LLNL.

While there were recent efforts to rely less on proxy apps and benchmarks and focus more on our restricted (export controlled) production-level applications, this work period should continue to move in that direction. Proxy apps are useful for identifying and isolating issues, but do not accurately capture the challenges that a production application will place on the software toolchain. With one year left until system delivery begins, approximately 18 months before our classified production applications are expected to run, and an increasingly robust software tool chain, we want to focus our efforts on the larger production-level codes to the largest extent possible, while using proxy apps for exploratory issues and reproducing bugs for off-site vendor perusal.

Early delivery hardware (Garrison nodes) are expected during this work period, and we anticipate relying on our IBM and NVIDIA partners in the COE to assist our code team build and test architects in bringing-up our applications on this new architecture, and assist in performance analysis, bottleneck identification, performance projections, and subsequent performance improvements for Sierra based on those projections. In particular, we need to continue to evaluate algorithms that may not be optimal to run on GPUs and work on multicore threading for the Power[8,9] CPU.

RAJA continues to grow in popularity as an effective abstraction layer for large ASC production codes. IBM and NVIDIA compiler support for RAJA idioms continues to improve, and several application teams are close to putting RAJA-enabled production code in their mainline development branches and "floor" versions of the code. As such,

continued ongoing support from IBM and NVIDIA toward improved compiler features and performance using RAJA techniques will be important. To the extent that LLNL can better make clear our approach and the evidence-based advantages of using RAJA, we can arrange for more detailed discussions with IBM and NVIDIA upon request.

A number of applications (such as Ares and KULL) will use Unified Memory (UM) for initial porting. We expect that CoE will help to estimate the effort required to restructure UM enabled codes to non-UM codes (codes with explicit use of the "map" clause for data motion).

In all cases below, we will reevaluate the work every two months and make adjustments as necessary.

2 Work Plan

Detailed project descriptions are described below. They are not in a priority order, but ordered similar to how they have been in past work plans for continuity.

2.1 Deterministic (Sn) Transport (Structured)

The structured Sn transport work performed to date with the *Kripke* proxy app has given the production application *Ardra* a solid path forward using techniques developed in Kripke. Ardra is currently undergoing a major incremental refactoring effort so that they can incorporate RAJA and other lessons learned in COE. This major effort is being performed by core members of their team, and is expected to take most of this work period to complete.

When this refactoring work is completed over the next year, we will reintroduce COE efforts to work with the Ardra team. In particular, the successes of OpenMP4 demonstrated in the last work period will need to be explored in the context of Ardra. The plan for them to use RAJA will allow for an easy transition between OpenMP4, CUDA, and potentially other programming models

For this work period, support from the COE vendors will be limited to addressing any issues with porting that code to the Garrison nodes, compiler issues, or other issues that come up related to applying the algorithms and lessons learned from Kripke in the context of a production application (unlikely).

Deliverables:
LLNL's Ardra team will brief CoE on the progress made in readying Ardra to Sierra and significance of the CoE work on Kripke.

IBM/NVIDIA target level of effort (%): 0%

2.2 Deterministic (Sn) Transport (Unstructured)

To date, unstructured Sn transport has been mostly explored in the context of the *UMT* benchmark application. While applicable (and important for the acceptance test benchmark suite), there have been some algorithmic changes to *Teton* (the production application upon which UMT is based) that may affect how easily explorations in the context of UMT can be transferred to Teton. As such, LLNL plans to focus significant attention on working directly on the Teton code base, and LLNL is committing significant resources over the next two work plan periods to ensure this – including direct attention from the primary author (Nowak), 50% support from an AAPS team member (Black), and an advisory role from a Teton user looking at the impact of domain decomposition on the solution time (Pearce). Likewise, we anticipate that the work Steven Rennich (NVIDIA) has done will transition smoothly into this work period, with better integration with the Transport team at LLNL working these issues.

Building on the prior work done in UMT, the memory demands and typical usage patterns of Teton will almost certainly require that we explore a "tiled" approach to implementing the computationally expensive sweep algorithm. LLNL has empirical evidence to show that shrinking the domain size down small enough to fit in GPU shared memory will have a negative impact on the iteration count, and thus the total solution time would likely increase even if the smaller domain size allow for a faster solve on the GPU per iteration.

Teton is largely Fortran-based, and as such will be one of our two major code efforts (Miranda being the other) we will use to help drive priorities with the IBM compiler team working on the xlF/OpenMP4 compiler. A goal for the end of this work period will be to have a well-defined path forward for Teton, and some initial prototypes demonstrated.

Assistance from Livermore Computing (e.g. Gyllenhaal, Earl) to install regular code drops for use in the LLNL RZ network is assumed, and will be coordinated with this effort and team.

Early in this work period, we will establish a small team focused on this effort to work toward building on prior effort and build a plan going forward. This will be followed by regular on-site meetings as needed, taking into account when our COE vendor partners are available on-site.

Note: **Only US Persons can access Ardra and Teton.**

Deliverables:
1) Exploration of tiled approach for problems not fitting into the GPU memory.

2) Exploration of IBM's Fortran compiler(s) supporting OpenMP4.5 for porting Teton to GPUs.
3) Developing a well-defined path forward for Teton, and some initial prototypes demonstrated.

Target dates:

- End of October 2016 – progress evaluation.
- End of December 2016 – progress evaluation.
- February 28, 2017: draft report submitted to LLNL

2.3 Hydro Codes

- **Ares and ALE3D**

Work on *Ares* and *ALE3D* is largely being undertaken directly by LLNL's code teams and supported by the *RAJA* project and team. The COE will require continuing effort for these codes to help streamline addressing compiler issues (e.g IBM and CUDA compiler support for RAJA, and OpenMP4.5 support for the IBM compiler). Additional help from effort here might be in doing some initial porting and performance tuning work on the Garrison nodes.

ALE3D has developed and adopted a memory management layer called CHAI to manage memory motion to and from the device without the need for explicit directives. It uses the lambda constructs available in RAJA along with a "managed pointer" type and small runtime to manage data motion. Ares is choosing to rely on Unified Memory (UM) to manage data motion, and as such – more profiling is generally required to understand and optimize data motion. Details of the work performed by these two teams is available in an ASC Level 2 (L2) milestone report which will be made publicly available at the start of this work period. LLNL suggests that a WebEx be set up to present this effort to our vendor partners once the L2 report has been accepted and approved. It is expected that final report will contain considerations for choosing UM over explicit data motion (using OpenMP4.5 directives).

- **Kull**

For Kull, we will begin work on transferring lessons learned in the *tinyHydro* mini-app into Kull. Members of the Kull team will be responsible for refactoring the code base and data layouts as suggested in the COE4 report, in collaboration with IBM (Grinberg) that evaluated tinyHydro in COE4. At a minimum, we will want to determine if Kull will perform better on the Power9 CPU or GPU, which will require performance projections once initial implementations (using either OpenMP4.5 or RAJA) are complete. Kull is also choosing to rely on UM to manage data motion. It is expected that final report will contain considerations for choosing UM over explicit data motion (using OpenMP4.5 directives).

- **Miranda**

Miranda will be participating in the IBM OpenMP4 hack-a-thon (September 2016) as an initial foray into threading and targeting GPUs. As the IBM OpenMP4.5 compiler technology matures to the point where Miranda can be compiled and targeting CPUs and GPUs, the COE will continue to build on that initial work from the hack-a-thon to define a path forward for Miranda.

- **Blast**

Blast is a hydro option based on high-order methods and relies heavily on the MFEM solver package (which is part of the ICOE effort). For this work period, we will monitor MFEM efforts in the ICOE (largely being undertaken by LLNL staff) and perform a profiling analysis to determine the approximate level of effort between Blast and MFEM. This will be done to define a plan for optimizing in future work plans. The plan of record is to port Blast using RAJA, so we will follow a similar path as was taken with ALE3D, Ares, and Kull: Initial profiling and bottleneck analysis, with a path forward defining efforts that need to be undertaken in MFEM and Blast for the follow-on work plan starting in Q11-12.

Deliverables: Deliverables will be defined at the end of October 2016, after initial porting effort and evaluation of IBM's Fortran compilers supporting OpenMP4.5.

Suggested IBM/NVIDIA target level of effort (%): 25%

Additional requirements: US Citizenship required to work on all of these production hydro codes.

Target deliverables and dates:

- End of October 2016 – progress evaluation and formulation of the Q9-Q10 deliverables.
- End of December 2016 – progress evaluation.
- February 28, 2017 – draft report submitted to LLNL

2.4 Material Libraries

One major looming question that needs to be addressed in the COE is how to best handle library calls to LEOS (equation of state), and MSLib (material strength). These are commonly used in the hydro codes described above, and may present a significant perturbation on lessons-learned to date with our hydro apps, which have largely used simple analytic Equation of State routines as a first cut.

2.4.1 LEOS/LIP

LEOS takes a relatively small amount of time in a typical hydro application (< 10% of the hydro time), but due to the use of large read-only lookup tables (typically several hundred MB per MPI rank), it is not necessarily a natural target for GPU acceleration. LEOS also makes use of polymorphic classes, which could prove to be difficult to deploy on GPUs without significant refactoring.

In addition to the large-memory tables, LEOS requires field data input from the hydro codes, and produces field data that is immediately used by other portions of the core hydro algorithms. If every call to LEOS requires significant amounts of data to be transferred back to the CPU for execution, this will likely hinder any gains observed by running the entire hydro package (with relatively modest memory requirements) entirely out of GPU memory.

LEOS relies on a lower-level library called LIP (Livermore Interpolation Package) for managing 2D table lookup and interpolation, and this is where most of the work/compute is done. By default, LIP pre-calculates a number of interpolation coefficients that allow for faster execution at the expense of greater memory usage (up to 16x over the raw table data). Options exist to calculate those coefficients on-the-fly, and may be part of the evaluation performed as a memory/speed tradeoff analysis. Unlike LEOS, LIP is available to work on in an unrestricted environment, and could thus be explored outside of the LLNL Restricted Zone (RZ) if necessary.

The COE effort will start by analyzing the pros and cons of several approaches. E.g.

1) Running LIP code on the GPU with all data tables loaded in GPU global memory. Explore compute/memory tradeoffs available in LIP to potentially save memory at the expense of more floating point operations.
2) Running LIP code on the GPU with data tables stored in CPU memory (either "pinned", or paged in on demand using UM)
3) Running LIP code on the CPU with data tables stored in CPU memory, and input/output fields remaining pinned in GPU memory
4) Running LIP code on the CPU with data tables stored in CPU memory, and explicit memory transfers or UM managing the movement of input/output fields in coordination with the rest of the hydro package.
5) Other approaches?

For option 1 (all data tables in GPU memory), one potential important optimization will be to store a single copy of all tables independent of the number of MPI ranks. Normally, each MPI rank has its own copy of the tables, although some prototype work has been done in the past to use several options for sharing these tables between MPI ranks that are running out of the same memory space. That initial work was targeted at BlueGene/Q and standard x86 Linux clusters, and will likely need some development to work in a GPU context.

Any of the hydro codes under study above can act as test "host codes" for these studies, with Ares likely being the best test candidate – as that project has some shared staff who also work on the LEOS team (Burl Hall).

The LEOS team is meeting in early October to plot a path forward.

Considering that LEOS is used by many of the hydro codes under study in the COE, the exploratory work on LEOS can be linked to readying those codes to Sierra. LEOS

shares a developer with the Ares team, so performing these studies in the context of Ares once standalone work is completed is recommended.

2.4.2 MSlib

MSlib represents a material strength library that generally executes the call tree on one element of data at a time with the parallelism exposed by the calling code. All of the input and output data is collected at the library entry point for a single element into a simple C struct. Aspects are somewhat similar to the structure of the Quicksilver/Mercury code where the code executed in parallel represents a "big kernel" of a potentially deep call stack.

The complexity of the code can vary greatly depending on the material strength model applied to each element – although in general elements that take a similar call path through the library will be grouped together. In some complex material models that use iterative techniques, the number of iterations can vary greatly element-by-element, thus making memory convergence difficult.

MSlib calls can additionally be complicated by the fact that elements will make calls to the LEOS equation of state library described above (one element at time).

Overall deliverables: Develop a path forward for porting material Libraries to the Sierra nodes.

Suggested IBM/NVIDIA target level of effort (%): 15%

Target dates:

- End of October 2016 – progress evaluation and formulation of the Q9-Q10 deliverables.
- End of December 2016 – progress evaluation.
- February 28, 2017 – draft report submitted to LLNL

Additional requirements: US Citizenship required to work on all of these production material libraries (except LIP, which is open source).

2.5 Monte Carlo Particle Transport

2.5.1 Quicksilver

Work on Quicksilver was initially undertaken in the last work period, and this work period will build on that effort, and on the lessons from the IBM's OpenMP4.5 hack-a-thon (September 2016).

In particular, in Q8-Q9 Quicksilver lite was used to investigate the viability of the "big kernel" approach that keeps the current coarse-level threading in place, with

each thread potentially taking divergent paths through the deep kernel stack. The performance of Quicksilver-lite has been sufficiently encouraging that we are willing to commit to a big-kernel GPU port of Quicksilver.

A GPU port of Quicksilver will force us to confront three main issues:

1. Quicksilver (and Mercury) make extensive use of work queues. For example, both the particle vault and MPI communication buffers can be viewed as work queues. We will need to identify performant, thread safe solutions to push and pop particles from queues.
2. We need a thread safe solution to incrementing tallies. Quicksilver and Mercury currently replicate all tally data for each OpenMP thread running on CPUs. This solution will not work for GPUs due to the very large number of threads that will be launched.
3. We will need a strategy to manage inter-node communication, especially particles that move across domain decomposition boundaries and the test for done algorithm. Initially we can adopt the simple strategy of buffering all particles that need to be sent to a neighbor node until all local particles have been processed, then exit the kernel, perform MPI communication on the CPU, and re-call the tracking kernel on received particles. However, we believe more advanced techniques will be needed to achieve desired levels of performance and scalability. Once Garrison nodes arrive we would like to start investigating GPU direct-based solutions.

Working on these issues in Quicksilver provides a more nimble code base to explore design options and also avoids Mercury's access restrictions. Moreover, Quicksilver is sufficiently representative of Mercury that the lessons learned can be applied to the design of Mercury.

2.5.2 Mercury

Mercury has done some significant threading work in the context of the Trinity COE. Our existing code uses the history-based approach, i.e., we track one particle at a time, for as long as we can on each thread (until the particle is terminated, reaches census or is buffered for MPI communication). LLNL plans on continuing this approach with the GPU using one-big-kernel, meaning each GPU thread will process a particle until it is terminated, reaches census, or must be communicated to another node. To keep the code base maintainable, it will be essential to minimize the code differences between tracking on the GPU and the CPU. Ideally, exactly the same code base will process particles for both the CPU and GPU with any necessary differences hidden behind abstractions.

One major aspect of Mercury that is not addressed in the Quicksilver mini-app is the use of complex cross-section libraries. Mercury can use both the MCAPM and GIDI libraries as options, but GIDI is the future path-forward, so we are not planning to

port MCAPM to GPUs. We do plan to work on a GPU port of GIDI as the start of an effort to understand how third party libraries will need to be modified to function in our one-big-kernel strategy.

For the duration of this work plan we believe that the majority of design exploration can be performed in Quicksilver. However, as time allows we may begin working on porting enough of Mercury to solve simple test problems on the GPU.

Because the history-based approach has been shown to be performant on the GPU using both ALPS (work of LLNL graduate student Ryan Bleile) and Quicksilver-lite, we do not currently intend to pursue event-based approaches. Converting Mercury to an event-based approach would require extensive code changes and would only be considered as a last resort.

Initial deliverables:
1) Addressing the three main issues as identified for Quicksilver
2) Porting Quicksilver and Quicksilver light to Garrison nodes
3) Porting GIDI to GPUs

Suggested IBM/NVIDIA target level of effort (%): 15%

Additional requirements: US Citizenship required to work on Mercury

Target dates:

- End of October 2016 – progress evaluation and formulation of the Q9-Q10 deliverables.
- End of December 2016 – progress evaluation.
- February 28, 2017 – draft report submitted to LLNL

2.6 Grand Challenge Project

LLNL is still finalizing plans for development of a "Splash App", and will engage IBM and NVIDIA as they solidify. Because of its open nature, likelihood of generating publishable material, and focus on performance – this may be an appropriate exercise for a new post-doc hire. We will determine that once the post-doc is on site. Should this effort command additional COE resources, we will adjust priorities as needed during the work period.

Level of effort: ~10% (to be adjusted as needed)

Target dates:

- End of October 2016 – progress evaluation.

- End of December 2016 – progress evaluation and formulation of the Q9-Q10 deliverables.
- February 28, 2017 – draft report submitted to LLNL

2.7 Collaboration with SNL and LANL

- LLNL, LANL, SNL, IBM and NVIDIA will define specific areas and deliverables for projects of interest for Sandia and Los Alamos.
- LANL is requesting to be invited/included in any workshops and hack-a-thons as observers, as well as a WebEx or VTC to go over the details of the Sierra node architecture
- SNL is requesting information and engagements specific to the following topics
    1. Compilers: robustness and performance (compile times, executable/library size, etc.)
    2. Math libraries: performance of ESSL/PESSL for small, non-square matrices
    3. Extended evaluation license for WSMP
    4. Performance of OpenMP tasking on the host. Especially their implementation of wait policies, task scheduling constraints etc.
    5. Solver (DD and MG) performance on the IBM/Nvidia architecture.

.

Level of effort (%): 10%

Target dates:

- End of October 2016 – progress evaluation.
- End of December 2016 – progress evaluation.
- February 28, 2017 – draft report submitted to LLNL.

2.8 Crosscutting Activities

As more of the Sierra preparation work transitions into the code teams for primary support, there are several COE activities that should be performed that will impact multiple projects.

Many do not include the "suggested staff %", as due to their cross-cutting nature and applicability to the projects described and accounted for above.

2.8.1 Porting to the Garrison node

Most internal code efforts to date have been performed on available x86/GPU clusters available. (Some have explored the Power8/Kepler nodes available on the rzmist node available at LLNL, but the lack of maturity of the toolchain and testbed nature of rzmist have made it a secondary target for application teams at LLNL.

It is anticipated that once the Garrison nodes come available in the first part of this work period that code teams will shift more of their effort to porting to the new architecture as part of the standard preparation for Sierra. It is likely that some training (or documentation) of this new environment will be needed, and that some assistance from IBM and NVIDIA will be required to optimize developer productivity in performing this port.

Deliverable:
The ORNL COE has requested training for their CAAR teams. LLNL would like to monitor or send a representative to this training to help determine if a similar training at LLNL would be valuable (e.g. via an on-site "hack-a-thon"). This would be most valuable if it were held on the LLNL RZ platforms using our more restricted production codes, which would require US citizenship (or "US persons") to access.

2.8.2 Performance Evaluation and Projection

As codes are ported to work on the GPUs, we must begin work on projecting performance to the Sierra node architecture based on experience with current x86 systems and the Garrison node early delivery systems. In particular, there is concern that bottlenecks related to data motion (either explicit or via UM) will be difficult to evaluate as hardware support for UM comes available with NVLINK2 on the CORAL node.

IBM team has a group working on putting together a projection methodology from Garrison to CORAL. As this effort is solidified within IBM research, we will apply this methodology to LLNL projects to a) project and predict performance to the CORAL platform, and b) validate the methodology as hard data becomes available in later work plans.

Deliverable:
LLNL requests that as this work is being developed within IBM, that it be presented to interested LLNL, SNL, and LANL staff (perhaps via a WebEx) with other follow-on efforts to be subsequently determined..

2.8.3 RAJA compiler support

As noted in the introduction, RAJA has become integral to our Sierra COE strategy for many of our application codes seeking performance portability and productivity. Some of the most impactful interactions we've had within the COE have been in closely working with the compiler teams to provide quick iterations on compiler features and fixes that were otherwise be show-stopping. As part of the overarching COE goals, we would like to continue to see strong coordination and cooperation between the applications and compiler teams in addressing these issues. The ability

for RAJA to allow teams to quickly switch between using OpenMP4 and CUDA when an issue arose with one of those compilers has been invaluable to ensuring continued forward progress. Being able to meet with the NVIDIA compiler team at their Santa Clara location was a hugely beneficial event. Likewise, having on-site representatives advocate for our needs has also been highly valuable.

LLNL is tracking a current list of bugs by severity and impact on our internal COE wiki page.

Deliverable:
We would like to consider having some regular telecons or WebEx sessions (e.g. every 4-6 weeks) with key members of the IBM and NVIDIA compiler teams as needed to track those issues. This would be either in addition to, or in coordination with, existing calls happening with the tools working group, but focused on application issues (specifically RAJA).
We understand that some issues are more difficult to solve than others, and two-way communication that can inform us of either a) why a particular bug or feature is either not feasible in short timeframes, or b) what some possible workarounds might be - would be highly beneficial. We would also request that Sandia and LANL be invited to these discussions (or that we merge with existing discussions they are already having), as many of the issues they have with Kokkos are similar in nature.

Suggested IBM/NVIDIA target level of effort: Occasional telecons + RAJA focus

2.8.4 IBM OpenMP4 Hack-a-thon follow-on

A second IBM OpenMP4 hack-a-thon is scheduled to be held Sept 13-16, 2016. The ASC Sierra COE will be sending two teams (Particle transport / Quicksilver and Miranda). The COE should take the lessons learned in the hack-a-thon to help clarify efforts for those codes outlined elsewhere in this work plan.

LLNL ASC COE staff attending:

- **Bill Cabot (Miranda)**
- **Dave Richards, Shawn Dawson (Quicksilver)**

2.8.5 Online GPU Training Outbrief

In the last work period, NVIDIA provided access to LLNL developers to Quicklabs online training courses. As a conclusion to that training, NVIDIA has offered a capstone event to be held at the LLNL HPCIC to answer detailed questions about CUDA, GPU programming, and specific application concerns related to the training.

Early October was suggested to attendees as a target date. Early in the work period, we will establish the date for this event, outline the specific goals and requirements for attendees (to be sent out several weeks in advance), and get an estimated number of attendees from the NNSA labs.

Suggested staff:

- **NVIDIA DevTech team (# TBD)**

Suggested NVIDIA target level of effort: One day at LLNL

2.8.6 COE-PP follow-on meeting

A multi-lab multi-vendor Centers of Excellence Performance Portability meeting was held in Glendale, AZ April 19-21. A final report will be available shortly on the meeting web site (https://asc.llnl.gov/DOE-COE-Mtg-2016/) and outlines some suggested paths forward for the DOE, including the possibility of a follow-on meeting. It is anticipated that hands-on experience with both Garrison nodes and Intel Knights Landing will provide some important lessons to be shared across the DOE ecosystem.

DOE is early in the process of deciding whether and when to hold a potential follow-on meeting. It is currently likely that LANL will lead the effort, and that a steering/planning committee consisting of representatives from each lab and vendor will be asked to participate in planning, potentially during this work period.

Suggested IBM/NVIDIA target level of effort: A few hours of telecons

2.8.7 Programming Strategies White Paper

The Sierra/Summit Programming Strategies White Paper is intended to be a living document. Some level of collaborative effort should be applied this work period to refresh this document, focusing on how the overall advice has evolved. Equal contribution from the LLNL and ORNL as well as IBM and NVIDIA is expected.

Note: This could make a good public document once the information goes GA after Sierra/Summit are released, or become the basis for portions of a RedBook effort at IBM to describe porting and optimization strategies.

Suggested IBM/NVIDIA target level of effort (%): 5%

2.8.8 Office hours and Unplanned Consultations

On site staff available for quick questions and consultations. It is understood that some “drop in” questions can be dispatched quickly, while others necessarily require staff to spend potentially multiple days exploring detailed options. For the latter, IBM and NVIDIA staff are encouraged to do what they can without significantly deviating from the work plan as even short consultations and suggested paths to explore can save significant effort for the developer. For more involved activities or requests, consult with the COE leads (Neely, Still) on how any

deviations should be handled or prioritized, perhaps involving a modification to the work plan.

Suggested IBM/NVIDIA target level of effort: about two hours/week.

3 Q9-Q10 work-plan adjustment.

LLNL, IBM and NVIDIA will review the progress and adjust the Q9-Q10 work-plan (if needed) every two month (at the end of October and December).