

RPA and libXC Meeting Minutes

Ideas for RPA Crash

- verify VBIOS settings same as nvidia (yes)
- try without "setenv CUDA_DEVICE_WAITS_ON_EXCEPTION 1" (still fails)
- run crash_test.c at keeneland (**fails there too**)
- change bios settings as recommended by colfax (still fails)
- swap C2075 with M2090 (**problem follows the M2090!**)
- remove IB card (still fails)
- try random matrix data instead of fixed data (**fails with random, works with fixed**)
- run colfax memory test
- reduce power consumption with "nvidia-smi --perf-limit=P4" (also tried P9). (**Fixes crashes, but still get corrupt data**).
- try 1 gpu per node, to see if gpus are "fighting", or if cooling/power is a problem (**still fails**)
- look for particular set of gpus that fail
- switch to cudamemcpy in crash_test.c
- make matrix bigger in crash_test.c
- run crash_test.c on 1 gpu per node (still fails)
- compare suncat-gpu-test (doesn't fail) and suncat-gpu (fails):
 - C2075 vs. M2090
 - 7 vs. 8 gpus
 - IB card
 - cooling
 - power cables? (c13 vs. c19)
- run nbody gpu test, as suggested by colfax (doesn't fail)
- read gpu temps (read out via ipmi: code fails with temp around 69C (lower than C2075 where nvidia-smi reports 88C))
- read rack temps (67-72 at inlet, 86-94 at outlet)
- run crash_test.py with P9 (no failures)
- run 32 gpu N-N with 3-gpus per node in exclusive mode (rack 1 still warm on the outputs: 88,88,92 (top to bottom on the front panel). Still saw nan's in dbgcrash_fast/try19.
- run with rack doors open, or change rack cooling behavior
- check that PyArgParseTuple types match between python/C
- cuda4
- check power
- small c version of crash_test.py
- look at ipmi errors on gpu24/26: nothing
- look in /var/log/messages for errors from driver
- security scans
- run gpu hardware tests (with colfax software?)
- gcc instead of icc
- small file crash (yes, crashed after 2 days)
- keeneland (saw 1 nan failure and 1 kernel launch failure)
- simple gemm test crash
- does it crash on 1 node? (yes on suncat-gpu 4 cores (gpu20, and gpu26), but not on suncat-gpu-test)
- mpi errors?
- try magma GEMM (still crashes)
- eliminate IB fork warning (still crashes)
- race condition between cublasDestroy/cublasCreate? (no, happens after first create)
- study with valgrind (dbgcrash_fast/try11,13,14 show some uninitialized data in mpisum)
- study with cuda-memcheck (see dbgcrash_fast/try12) looks clean for the 32-node job, even when the data gets messed up (see many warnings about python numerical overflows, indicating failure has occurred)
- understand imprecise exceptions
- run @nvidia with 8*M2090, Tyan motherboard, and cuda 5.0 (**works!**)
- hardware problem (check for common node, too many jobs in the logfiles)
- read the code: cudamemcpy memory overrun?
- discontinuous numpy array? (put in asserts)
- did get a memory error when running racecheck
- no errors from cuda-memcheck heap check
- read code to look for race conditions in cukernels.cu (even though problem existed before the addition of those kernels)
- ran cuda-memcheck racecheck: only saw errors from cublas
- check ecc enabled. looked with "nvidia-smi -q"
Ecc Mode
Current : Enabled
Pending : Enabled

To-Do List

Nvidia GTC questions

- what intermittent errors does cuda-memcheck not detect?
 - hardware
 - cudamemcpy
 - others?
- cuda-gdb generates output for kernel launches. slows down the code dramatically? becomes unusable.
 - set flag "set cuda kernel_events 0"

- submit bug report if not solved
- how does cuda deal with memory fragmentation?
- nvvp error: "102 metrics have invalid values due to inconsistencies in the required event values"
 - trying to match up the counters in time, if not well-synchronized gives the above error. Larger pages.
 - handled differently by nsight (replays previous profiler)
- double complex math: really fp64 instructions?
 - only double-precision
- talk to Gernot Ziegler about instruction limited kernels?
- is our zherk kernel latency limited?
 - multiple of 8 for k (8 rows at time in the loop)
 - may be limited by pieces at the beginning/end (end: scaling by alpha, beta, beginning: load the shared memory) loop over k in the middle
 - kepler: k up to 1000 for top performance
- cufftplan many memory leak
- trigger crash on nan? how do nan's get produced?
 - not possible to trigger a crash on nan
- why do they use bytes-per-instruction
- get many errors from cublas with race check
 - if really errors: submit bug report
- if we have 1 number used by many threads should it go into shared memory? constant memory?
 - we would think constant memory would be the right answer. shared memory would give a bank conflict.
- what memory access errors can memcheck detect? cudamemcpy? array-out-of-bounds?
 - doesn't detect cudamemcpy errors (or any errors by the host) but does detect array-out-of-bounds accesses within the GPU
- we were not 16-byte aligning cuDoubleComplex variables. error showed up "much later" in cuGetVector (error 11) and cudaDeviceSynchronize (error 4). Did binary search to find source of error. How do we program error-checks so that run-time errors show up "immediately"?
 - cuda_safe_call?
 - pattern for error checking: issue different kernels in different streams, then do cudastreamsynchronize and cudagetlasterror
- understand crash with rpa-gpu-expt running rpa_only_Na_cuda.py with nvprof
 - should file a bug report

3/12/2013

- look profiling on RPA (lin)
- ask about error handling at GTC (lin)
- base.py get_phi_agp kernel
- rpa manuscript (jun)
- k-point parallelization (cpo)

3/5/2013

- 2 slides for Samuli
- run profiling on RPA (lin)
- memory leak (perhaps related to crashes)
- adding error check functions
- base.py get_phi_agp kernel
- rpa (jun)
 - manuscript

2/26/2013

- 2 slides for Samuli
- more structs for RPA (lin)
- run nvvp on RPA (lin)
- think about EXX
- rpa (jun)
 - manuscript

2/19/2013

- more structs for RPA (lin)
- commit code
- rpa (jun)
 - keep on eye on crashes
- EXX on GPUs
 - fix MPI stuff in EXX
 - understand why it doesn't speed up
 - think about whether or not we tackle EXX yet

2/12/2013

- structs for RPA
- rpa (jun)
 - keep on eye on crashes
 - EXX on GPUs

2/5/2013

- profile GPU-GPAW with maxed out memory on the GPU (lin)
- gpu-gpaw profiling thoughts:
 - cpo thinks improving mpi performance may be difficult
 - lin thinks improving mpi performance may be important, since number of k-points decreases in future.
 - maybe we could pipeline other work while mpi is running?
 - why do we only get a x5 speedup for Pt 3x3x4? (samuli sees 8 to 11)
- see if the mask stuff is called every SCF step (aj)
- think about randomization idea (aj)
- evaluate effectiveness of tzp+PK on na2o4/pt (aj)
- freeze D_aps?
- rpa (jun)
 - keep on eye on crashes
 - rewrite code for the ZHERK

1/29/2013

- profile GPU-GPAW in grid mode (lin)
- gpu-gpaw profiling thoughts:
 - domain decomposition is especially inefficient on GPU: pack as much domain onto one GPU as possible (need larger memory)
 - parallelization over k-points remains good
 - our current 1-k-point on 8 cores is unrealistic for a 3x3x4
 - cpo thinks improving mpi performance may be difficult
 - lin thinks improving mpi performance may be important, since number of k-points decreases in future.
 - maybe we could pipeline other work while mpi is running?
 - why do we only get a x5 speedup for Pt 3x3x4? (samuli sees 8 to 11)
- email the list about the real-density mixer (aj)
- see if the mask stuff is applied every SCF step (aj)
- think about randomization idea (aj)
- evaluate effective of tzp+PK (aj)
- rpa (jun)
 - keep on eye on crashes
 - rewrite code for the ZHERK

1/22/2013

- libxc on gpu (lin)
 - work on automake stuff on Thursday
 - ping Miguel
- AJ tries simple new-setup Pt system with rmm-diis
 - use same code with different setups or vice-versa
 - generate residual compared to converged
- cpo compares FFTMixer to dacapo
- rpa (jun)
 - merge trunk and print pointers to understand crashes
 - rewrite code for the ZHERK

Questions for Nvidia

- how to use constants memory
 - constants memory: broadcast same 4 bytes to all threads of a warp, if the request is synchronized. a performance penalty if they don't. must explicitly call it out with `_constant_` (for kepler "immediates" are stored in the constants memory if large enough, otherwise instruction).
- how to use texture memory
 - textures: if used in a "2D or 3D" manner, can only store 4 bytes. in kepler: can use ldg. memory has been ordered in a strange way ("snake") to allow better accesses to multi-dimensional stuff. ugly with double precision, because of the 4-byte size.
- what does the 150GB/s mem bandwidth number mean?
 - it is sum of read/write bandwidth (each is 75GB/s)
- optimization tricks: pre-fetch etc.
 - we get 85GB/s out of 150GB/s on 2075. use cudaDMA?
 - philippe measures 84% memory bandwidth (154GB/s) on K20
- what does a queued warp do? (does it pre-fetch the memory)
 - yes, but can do better (e.g. cudaDMA)
- reducing number of registers in kernel (does compiler typically do this optimally?)
 - can control register usage using launch bounds
- how to learn with nvvp if we're memory/flops limited
 - philippe just counts instructions and measures MB/s by running code (no NVVP). He has some special code that counts instructions for him in complicated cases.
- understanding the nvvp columns
 - ECC decreases memory performance 20%. (118GB/s for 2075)
 - 106GB/s is "quite good"
 - 90% is the highest
 - maybe we should turn off ECC? Will lose statistics.

dynamic shared memory: extra launch parameters: stream and amount of shared memory to allocate dynamically

instruction replay overhead:

- "sum" (different columns have different denominators) of next 3 columns:
 - o can replay because needed to fetch multiple cache lines per global memory access instruction (e.g. because of cache-line misalignment)
 - o can replay because needed to fetch multiple cache lines per local memory access instruction (NOTE: this is "LOCAL MEMORY CACHE REPLAY OVERHEAD")
 - o shared memory bank conflict
- global memory store efficiency: measure of stored bytes vs. "real" stored bytes (should only be <100% if we have cache-line misalignments)
- local memory overhead: measures local memory accesses (stack traffic, register spill traffic)
- warp execution efficiency: measure of branch-divergence (percentage of threads that are active in a warp)
- global memory load efficiency: measure of loaded bytes vs. "real" loaded bytes (should only be <100% if we have cache-line misalignments)
- achieved occupancy: this is from "tail" from the numerology of number of waves of blocks
- instructions issued: number of warps instructions issued to all SMS. compare to 1.15 GHz*#SMS*duration (maximum of 1)

NOTE internally fermi: really runs 2 half-warps over 2 clocks, but the above math still works out for the simple-minded.

NOTE: executed: first time , issued: includes replays

- best way to associate right GPU with right core (e.g. "taskset", "numactl")
 - if numactl settings OK, OS should take care of that. still have to correct taskset/cuCreate
- ask about zher speedup numbers: for 4kx4k why does gemm improve by x30 but zher improves by x6?
 - gemm with large sizes is compute limited, which GPU does well. zher is memory limited.
- using automake with cuda and c in one library?
 - no good answer
- nvidia-proxy allocation: free up memory?
 - proxy doesn't provide a good way to free up memory

1/8/2013

- libxc on gpu (lin)
 - work on automake stuff
 - get the cleaned-up ifdef version from Miguel
- digest RPA timing measurements (lin)
- AJ tries simple new-setup Ru system with rmm-diis
 - generate temperature residual plot
 - generate residual compared to converged
- cpo compares FFTMixer to dacapo
- paper (jun)
- redo timing measurements (jun/lin)
- understand new GPU box memory slowness (cpo)

12/18/2012

- libxc on gpu (lin)
 - use common work file for CPU/GPU
- digest RPA timing measurements (lin)
- paper (jun)
- redo timing measurements (jun)
- understand timing measurements more fully (jun)
- dacapo density mixing vs. GPAW (cpo)

12/11/2012

- understand nvidia zgemm speedup plot (jun/cpo)
 - ANSWER: without thread: 29 faster on GPU. With 6 thread openMP get 5, which agrees with nvidia
- understand why zher is x6 better on GPU but we see x24 with RPA (will put device sync in code) (jun/cpo)
 - ANSWER: CPU is memory bandwidth limited (so faster with 1 core). account for roughly x2, and the other x2 comes from overlapping CPU/GPU computation.
- does cuda5 improve ZHER? (jun/cpo) ANSWER: no improvement
- libxc on gpu (lin)
 - use common work file for CPU/GPU
- digest RPA timing measurements (lin)
- think about moving lambda calc to GPU (jun) (ANSWER: no need, 10 or 20% improvement, best case)
- try multiple surfaces with jacapo/gpaw-pw (aj)
- paper (jun)

- try calling dacapo density mixing from GPAW (cpo)
- make sure all libxc self-tests run
- why doesn't marcin's na.py converge, even with fixed density?
- can the alphas for the nt_G really be used for the D's?

12/4/2012

- understand nvidia zher speedup plot (jun/cpo)
- libxc on gpu (lin)
 - use CUDA5
 - use common functional file for CPU/GPU
 - use common work file for CPU/GPU
 - read samuli old talk
 - run 3x4x3 pt system
- RPA timing measurements (lin)
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- try multiple surfaces with jacapo/gpaw-pw (aj)
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)
- make sure all libxc self-tests run
- move suncatgpu01 to CUDA5 (cpo)

11/27/2012

- come up with list of items to ask about at nvidia mtgs
- libxc on gpu (lin)
 - read samuli old talk
 - run 3x4x3 pt system
 - run PBE0
 - fix linking undefined symbol
 - make sure all self-tests run
 - put paramsize fix in for mgga and lda
 - test libxc 2.0.0
- RPA timing measurements (lin)
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- try multiple surfaces with jacapo/gpaw-pw (aj)
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)
- "patch" file for libxc (only the memsets?) (cpo)
- move suncatgpu01 to CUDA5 (cpo)
- figure out how to softlink lda_c_pw.cuh (cpo)

11/20/2012

- libxc on gpu (lin)
 - fix the zeroing (is there a cudamemset?)
 - make sure all self-tests run
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- try multiple surfaces with jacapo/gpaw-pw (aj)
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)
- merge libxc-gpu and libxc (patch memsets, and zero-ing in work) (cpo)
- "patch" file for libxc (only the memsets?) (cpo)
- move suncatgpu01 to CUDA5 (cpo)

11/13/2012

- libxc on gpu (lin)
 - fix the zeroing (is there a cudamemset?)
 - double check timing for LCAO results
 - make sure all self-tests run
 - commit to svn
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)
- merge libxc-gpu and libxc (patch memsets, and zero-ing in work) (cpo)

11/6/2012

- libxc on gpu (lin)
 - decide what to do about the hacks (with print statements)
 - copy less of the scratch data to GPU
 - run the self-tests
 - see if performance is better/worse
 - check that unmodified libxc still works
 - commit to svn
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)
- merge libxc-gpu and libxc (patch memsets, and zero-ing in work) (cpo)

10/30/2012

- libxc on gpu (lin)
 - remove print statements
 - merge libxc-gpu and libxc
 - copy less of the scratch data to GPU
 - run the self-tests
 - do the memsets for lda/mgga
 - see if performance is better/worse
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- install GPAW on Keeneland (cpo)

10/23/2012

- libxc on gpu (lin)
 - remove print statements
 - test spin-polarized
 - understand why H numbers are different than gpugpaw_v2
 - merge libxc-gpu and libxc
- multi-alpha zher at a lower priority(jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- try calling dacapo density mixing from GPAW (cpo)
- get journal recommendations from Nichols (cpo)

10/4/2012

- libxc on gpu (lin)
 - PBEsol-X
 - put libxc in samuli branch at "low-level" (libxc.py?)
 - solve zero-ing problem and stride problem
- multi-alpha zher (jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- create infrastructure for running convergence tests (aj)
- try calling dacapo density mixing from GPAW (cpo)

9/25/2012

- libxc on gpu (lin)
 - test PBEsol
 - cleanup existing code (delete commented lines, unused code)
 - put in p_d_gga and p_d_mgga, for consistency
 - have 1 beautiful program that runs a lda/gga/mgga functional on both CPU/GPU and times them.
 - think about integrating with samuli
- multi-alpha zher (jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - work on understanding jacapo density mixing

9/18/2012

- libxc on gpu (lin)
 - focus on tpss_x (summarize pattern for moving functional to gpu)
 - ask samuli if there are functionals he would like us to move?
 - figure out how to get nested param-size (will change "p" struct for this, in general it would be a function to deep-copy params)
 - figure out how to get p_d into the functional (will change "p" struct for this)
 - kinetic functionals
 - understand PBE instruction replays and constants-memory
 - think about cleanup of p
 - summarize pattern for moving functional to gpu
 - better pattern for p_d?
 - think about integrating with samuli
- multi-alpha zher (jun)
 - reduce registers? prefetch?
 - explore the parameter space: tile-size
- paper (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - work on understanding jacapo density mixing

9/5/2012 and 9/12/2012

- libxc on gpu (lin)
 - do mgga (summarize pattern for moving functional to gpu)
 - figure out how to get nested param-size (will change "p" struct for this, in general it would be a function to deep-copy params)
 - figure out how to get p_d into the functional (will change "p" struct for this)
 - kinetic functionals
 - understand PBE instruction replays and constants-memory
 - think about cleanup of p
 - summarize pattern for moving functional to gpu
 - better pattern for p_d?
 - think about integrating with samuli
- multi-alpha zher (jun)
 - run nvvp
 - look at occupancy calculator (get registers from nvvp)
 - think of new ideas to speed-up
 - explore the parameter space: threads-per-block, tile-size
- paper (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - work on understanding jacapo density mixing

8/28/2012

- libxc on gpu (lin)
 - do mgga (summarize pattern for moving functional to gpu)
 - figure out how to get nested param-size (will change "p" struct for this, in general it would be a function to deep-copy params)
 - figure out how to get p_d into the functional (will change "p" struct for this)
 - kinetic functionals
 - understand PBE instruction replays and constants-memory
 - think about cleanup of p
 - summarize pattern for moving functional to gnu
 - better pattern for p_d?
 - think about integrating with samuli
- multi-alpha zher (jun)
 - understand current code
 - understand nvidia suggestions
- fix timing of cublas vs. source-code zher and run benchmark
- paper (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - work on understanding jacapo density mixing

8/21/2012

- libxc on gpu (lin)
 - performance plot for RPBE (lin)
 - do mgga (summarize pattern for moving functional to gpu)
 - understand crash for large number of grid points
 - figure out how to get nested param-size (will change "p" struct for this, in general it would be a function to deep-copy params)
 - figure out how to get p_d into the functional (will change "p" struct for this)
 - read thru func_aux
 - kinetic functionals
 - time PBE
 - look at nvvp to understand bottleneck
 - think about cleanup of p
 - summarize pattern for moving functional to gnu
 - better pattern for p_d?
 - think about integrating with samuli

- multi-alpha zher (jun)
- paper (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - add Na₂O₄ calculation to AJ infrastructure
 - understand default jacapo/gpaw parameters/algorithms/initial-values

8/15/2012

- libxc on gpu (lin)
 - performance plot for RPBE (lin)
 - work on either the mgga or the copying of "p"
 - understand crash for large number of grid points
 - read thru fund_aux
 - time PBE
 - look at nvvp to understand bottleneck
 - think about cleanup of p
 - summarize pattern for moving functional to gnu
 - better pattern for p_d?
- evaluate possible gpu purchase (jun)
- multi-alpha zher (jun)
- paper and speeding up more (FFT?) (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - add Na₂O₄ calculation to AJ infrastructure
 - understand default jacapo/gpaw parameters/algorithms/initial-values

8/8/2012

- libxc on gpu (lin)
 - performance plot for RPBE (lin)
 - work on either the mgga or the copying of "p"
- evaluate possible gpu purchase (jun)
- multi-alpha zher (jun)
- paper and speeding up more (FFT?) (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - add Na₂O₄ calculation to AJ infrastructure
 - understand default jacapo/gpaw parameters/algorithms/initial-values

7/11/2012

- libxc on gpu (lin)
- evaluate possible gpu purchase (jun)
- multi-alpha zher (jun)
- create infrastructure for running convergence tests (aj)
- help with all the above (cpo)
 - add Na₂O₄ calculation to AJ infrastructure
 - understand default jacapo/gpaw parameters/algorithms/initial-values

6/27/2012

- libxc on gpu (lin)
- more convergence test cases (aj)
- think about FFT cutoff (aj)
- xsede machines
 - generate benchmark strong-scaling plots for exx/rpa for forge (jun)
 - create proposal rough draft (jun)
- finish libxc (cpo)

6/20/2012

- libxc on gpu (lin)
- more convergence test cases (aj)
- think about FFT cutoff (aj)
- xsede machines
 - install software on forge (cpo)
 - generate benchmark strong-scaling plots for exx/rpa for gordon/forge (no swapping!) (jun)
- finish libxc (cpo)

6/13/2012

- libxc on gpu (lin)
- more convergence test cases (aj)
- think about FFT cutoff (aj)
- xsede machines
 - install software on forge (cpo)

- understand gordon error (cpo)
 - generate benchmark strong-scaling plots for exx/rpa for forge (no swapping!) (jun)
- finish libxc (cpo)

6/13/2012

- try libxc on gpu (lin)
- more convergence test cases (aj)
- think about FFT cutoff (aj)
- see if we get 50% speedup with new zher code (jun)
- xsede machines
 - install software (jun/cpo)
 - generate benchmark strong-scaling plots for exx/rpa for forge (no swapping!) (jun)
- work on libxc (cpo)

5/30/2012

- understand x/c kernel bottleneck with nvvp (lin)
- trying cufft to see what we gain (lin)
- more convergence test cases (aj)
- think about FFT cutoff (aj)
- GEAM, ZHERK (jun)
- xsede machines (jun/cpo)
 - generate benchmark strong-scaling plots for exx/rpa (no swapping!)
 - use std err to look for node-to-node "time variations"
- work on libxc (cpo)

5/23/2012

- understand x/c kernel bottleneck with nvvp (lin)
- trying cufft to see what we gain (lin)
- use VO as convergence test case (aj)
- look at special-metric-weight convergence (aj)
- think about FFT cutoff (aj)
- GEAM, ZHERK (jun)
- build on hopper and xsede machines (jun/cpo)
 - generate benchmark strong-scaling plots for exx/rpa (no swapping!)
 - use std err to look for node-to-node "time variations"
- work on libxc (cpo)

5/9/2012

- rpbe kernel (lin)
 - does memcpysync need cudamallochost?
 - fix stream behavior and try with 1,2,4,8,16 streams
 - understand stream behaviour with nvvp
- zher streams(jun)
 - in benchmark, have separately variable nstream/nw
 - can we see whether we have 4 or 16 streams?
 - understand stream behaviour with nvvp
- density mixing (aj)
- work on libxc (cpo)

5/2/2012

- looking at EXX bottleneck (rewriting) (jun)
- use cuda streams for small RPA systems (jun)
- libxc integration (cpo)
- understand MKL benchmark (jun/cpo)
- pycuda (cpo)
- understand RPBE kernel: (lin)
 - understand "double" problem
 - vary np, block_size, nstreams
 - loop testfunc many times
 - longer term: look at jussi/samuli kernel for ideas

4/25/2012

- looking at EXX bottleneck (rewriting) (jun)
- postpone work on ZHER stuff until we have news from INCITE (jun)
- talk to Frank about computing time applications (cpo)
- understand MKL benchmark (jun/cpo)
- libxc integration (cpo)

4/18/2012

- look at reduced-scope libxc example plus RPBE (lin)

- if there is time, benchmark the RPBE kernel (lin)
- zher performance improvement with multiple streams (jun)
- make INCITE version work (jun/cpo)
- move to libxc 1.2 (cpo)

4/11/2012

- libxc parallelization (lin)
- libxc integration (cpo)
- understand missing time in cublas mode (jun/cpo)
- how to put the gemm in PW mode in a fairly neat way (lin/cpo)
- start working on multiple-alpha kernel (MAZHER) (jun/cpo)
- work on INCITE proposal (jun/cpo)

3/28/2012

- gemm (lin)
- run pt3x3 (cpo)
- libxc (cpo, and lin if he finishes gemm)
- cher/fft (jun)
- fix gpu allocation (cpo)
- circular dependency problem with monkhorst_pack (cpo)
- mpi failure with cuzher (cpo)

3/21/2012

- batch queue for GPU machine (cpo)
- fft/gemm/gemv (lin/jun/cpo)
- single precision cher instead of zher? (jun/cpo)
- new libxc (cpo)
- fix libfftw detection (cpo)
- improve zher in cuda (long project, jun/cpo)
- move "expand" from python into C, post to mailing list? (lin)
- look at spin paired (cpo)
- run pt3x3 (cpo)

3/14/2012

- pycuda compatibility (cpo)
- private svn (cpo)
- try nvvp/transpose (or C60 with more grid points) for >5 minutes (lin)
- send mail to nvidia or list to understand why nvvp profile cuts off after 5 minutes (lin)
- understand bottleneck in get_wfs (jun)
- implement fft/gemv (cpo)
- is there a cuda library for trace like zgeev (cpo)
- run a 3x3x3 system to see if bottlenecks stay the same (cpo)
- driver hang status (cpo)
- understand how to fix gs.py bottlenecks in more detail (lin/cpo) using gpaw profiler:
 - pseudo density: density.py: self.calculate_pseudo_density(wfs) (cpo)
 - projections: overlap.py: wfs.pt.integrate(psit_nG, P_ani, kpt.q) (cpo)
 - RMM-DIIS: eigensolvers/rmm_diis.py: lots of lines (cpo)
 - projections: eigensolvers/rmm_diis.py: wfs.pt.integrate(dpsit_xG, P_axi, kpt.q) (lin)
 - calc_h_matrix: eigensolvers/eigensolver.py: H_nn = self.operator.calculate_matrix_elements, hamiltonian.xc.correct_hamiltonian_matrix (lin)
 - rotate_psi: eigensolvers/eigensolver.py (lin)

Accessing suncatgpu01 SVN

We have put a version of GPAW in a local SVN repository on suncatgpu01. To access it, use the following:

```
svn co svn://localhost svngpaw
```

You can put whatever you want for the last argument (local directory name).

General Topics

- Stanford CUDA course: <http://code.google.com/p/stanford-cs193g-sp2010/>
- (Everyone) Understand gpaw (read paper)
 - what other steps could we parallelize?
 - Can we do existing parallelization better? (e.g. use ideas in Todd's GPU papers)
- (Everyone) Go through CUDA tutorial [here](#).
 - Understand blocks/threads/warps and how they map onto GPU hardware (details of which can be seen with "deviceQuery" command)
- (Lin) Find tool to measure:

- memory bandwidth usage
 - gpu flops usage
- (Jun) :
 - Parallelize LCAO/planewave/RPA (zher performance?)? non-rmm-diis eigensolver?
 - merge with trunk?
- (cpo) :
 - Understand code flow
 - Understand where the ~23 cuda kernels are used
 - Understand which bottlenecks we need to tackle
- Do another gpu-gpaw install (to learn)
- Understand Pt 3x4x3 CPU/GPU difference versus 3x2x3 (performance scaling with system size)
- Can multiple CPU processes win by using the same GPU?
- Understand pycuda
- Understand gpaw interface to cuda (c/cuda subdirectory)
- Read CUDA programming manual [here](#).
- Do all gpaw self-tests pass with GPUs?
- Can we get bigger bang-per-buck with GeForce instead of Tesla? (don't need GPUDirect, maybe live with less memory/bandwidth? double precision worse)
- Understand cuda better:
 - Does Samuli use pinned memory correctly?
 - run/understand cuda a bandwidth benchmark
 - Could we use GPUDirect for MPI data transfer?
- Does the GPU performance scale with the product of gridpoints*bands? Might be a combinatorial effect with the bands, linear with the grid points?
- Duplicate Samuli results
- Update to most recent version in svn
- Understand where gpaw scaling maxes out for Pt 3x4x3
- Why is CO on 2 GPUs slower than on 8 CPUs?
- Can we do something less precise in the vacuum area? (fewer grid points?)
- Do we need a fatter interconnect for GPUs?