Q/A Sheet: Node-Level Performance Engineering

1	Question	Answer
2	Ex. How do I download the tutorial materials?	Emails have been sent out with links where you can download the presentation notes and examples.
3	Could someone type out the download URL again? Thanks.	http://tiny.cc/NLPE-SC20
5	I am interested in knowing the Flops rating for AMD FP64 vs NVidia A100 F64?	Performance FP64 for AMD MI50 is approx 6.7 TF/s which is approx 1/3 of A100
6	What are the benefits of running in Sub NUMA Clustering mode?	Benefits: a little less latency and a little more saturated memory bandwidth. Downside: more complex node topology, you have to deal with the NUMA structure.
7	What new performance features do you see coming for the next generation of cpus?	There are three ways to speed up stored program computers: Increase clock speed (the gold option, everything gets faster, still alive with turbo mode), going parallel (the main driver in the last decade with SIMD and multi-core), and adding special purpose hardware. Of course we cannot look in the future but my guess is we will see more special-purpose solutions in the future. This can be special-purpose execution units or coprocessors.
8	Filling the pipeline seems to be key to maximizing performance, but many algorithms rely on branching and dependencies. Are there situations where using a less 'efficient', non-branching algorithms is more efficient on modern processors? Or perhaps techniques for reducing branching?	Complicated question. In many cases, the more efficient (i.e., work saving) algorithm is harder to optimize for the architecture but still better overall in terms of time to solution. But there are exceptions where algorithms with more work can actually solve a problem faster because they are better adapted to the hardware. Example: https://dx.doi.org/10.1137/140976017
9	Do you have any notes on type of applications workloads can benefit NUMA-feature NPS at boot time?	NPS != 1 will provide a little more memory bandwidth. Hence, memory-bound applications will benefit if they use proper page placement, i.e., if (almost) all memory accesses are to the local domain.
10	Will we have access to this recording? Having poor streaming issues on my end.	Yes, a recording will be available on Wednesday (I think), and you can view it for 6 months as an on-demand stream.
11	Is likwid-pin cpuset-aware?	If it detects a cpuset, it will use "logical pinning with the set," i.e., it will pin threads to consecutive hardware threads within the set. So yes, it is cpuset aware, but you'll get most out of it when the whole node is yours.
12	Does likwid-pin have a mechanism to skip placement of lightweight (non-compute) threads?	Yes. There is the "-s" option with which you can specify a "skip mask." A set bit in this mask denotes a thread to be skipped, i.e., not pinned.
13	On machines where you have more than is needed cores to handle a fixed work load, does it make sense to disable cores in order to maximize overclocking?	I think my answer was not 100% to the point. Of course, on modern, turbo-enabled multicore CPUs you can use fewer cores with higher frequency because there is more headroom in the power envelope. Now the really interesting question is, how do you burn the least energy without compromising time to solution? That's a complicated question because it depends on the exact way the turbo frequencies are set by

		the processor. Usually, if the workload is scalable across cores, there is an optimal clock speed for minimum energy to solution. If you want minimum time to solution, just use all cores and crank it up to the limit. If you _really_ just want m out of n cores, and energy is not at all important, then yes, use m cores, pin your threads, and leave the others idle to go into some power-saving state.
14	Is there a way to avoid OpenMP barrier or say reduce the barrier performance impact?	The way to reduce the barrier impact is to avoid barriers :-) Seriously, indeed it can be a challenge to fight barrier overhead. There is no silver bullet except to try and think about how you could move more work between successive barriers to reduce their relative cost. Sometimes it can help to think more in terms of tasks instead of parallel loops, but this is by no means a general strategy.
15	What are the reasons that the CPU vs GPU peak performance is only 4-8x? Is not being able to parallelize workloads to so many threads one of major reasons?	This is the ratio as given by the capabilities of the hardware; you see a similar ratio when running LINPACK. You do need more parallelism to get there on the GPU because it has no automatic latency-hiding mechanisms (like deep out-of-order execution or prefetching).
16	Could you give a very brief example where you would want to use the outer level cache group thread domain in LIKWID?	Sometimes the code needs a certain amount of cache to work optimally (as is the case for stencil codes and also for sparse matrix-vector multiplication, which will both be covered in the second part of the tutorial). Then you'd want to restrict the number of threads running per LLC domain. Also it's useful for microbenchmarking if you want to, e.g., look at the LLC bandwidth.
17	What would you suggest for digging deeper into measuring the performance relationships between the CPUs and other node PCI devices, such as high-speed (>= 40-Gbps) network interface devices (to better understand the bottlenecks impacting bandwidth-bound applications)?	This is a typical case for microbenchmarking. Write a simple benchmark that mimics the way your real application uses a data path. The insight thus gathered can then be used to understand the behavior of real applications. If you know how long certain things take, you can find out which ones are bottlenecks.
18	In the likwid-pin demo using the bandwidth test, at the higher core counts the second column of bandwidth results was sometimes significantly lower than the first column, why was that?	I'm not sure I understand the question correctly. The second column in the bwbench output is the MFlop/s number, and it's related to the bandwidth via the intensity (flops/byte). Probably you mean that some benchmarks (such as copy) seem to have lower bandwidth than others (such as update). This is because copy has a write-allocate that uses 1/3 of the bandwidth, but the benchmark has no way of knowing whether a write-allocate actually happens. This is why loops with (relatively speaking) more stores appear to have lower bandwidth, but if you add the write-allocate then all of them have roughly the same bandwidth.
19	Is it always safe to sample system-wide even when the application is instrumented internally, or does concurrent use of these performance counters by multiple agents	If by "safe" you mean that you can rely on the results to be correct, then you should use the "perf" backend, which works per process so that system monitoring will not interfere with your measurements. If you use the direct MSR access, you may get problems.

	impose any risks which need to be mitigated?	
20	how do you ascertain that you don't incur cache capacity misses (other than by experimetation)?	The analysis assumed the minimum data traffic that is theoretically possible. If the performance is then close to the Roofline prediction, this is already an indication that the actual data traffic is close to the minimum. Using performance counters (via likwid-perfctr, for example) you can measure the traffic directly and further validate the model. As mentioned in the talk about performance counters, we usually don't think in terms of cache misses but in terms of resource utilization.
21	Did you compare the example versus the case where M and N were not known at compilation time? Curious as to how much slower it would be.	Yes - depending on the parallelization and the compiler version and switches performs penalties of 10x and higher show up for the "general" version. To do code optimization (mood unrolling, interchange) and SIMD vectorization, the compiler must make assumptions about the loop length - if not known - and mostly fails to produce appropriate code for our parameter regime. As seen from the comparison with mkl also the vendor library is not well prepared for that parameter space. Automatic code generation is useful (and actually used) to build problem specific libraries in this application.
22	so roughly how much of the improvement was due to the telling the compiler the values of N and M vs the new way of parallelizing the K loop?	I do not have exact numbers here but if you chose a different parallelization strategy (loop for parallelization) scalability is strongly limited by the short dimensions. On the other hand knowing the value of M and N boosts single thread performance. So basically both effects multiply until you reach the roofline limit when increasing the number of execution threads. If you only apply one of the "optimizations" you will not be able to saturate the memory bandwidth and reach the roofline limit.
23	Is there any reason not to use nontemporal stores for the stencil example?	You are absolutely right. This can be done / enforced and will reduce the overall code balance by 8 B/LUP. On the single core you will thus not see a strong performance increase - on older Intel architecture you will even get worse single core performance. However, for the OpenMP parallel code - saturating the memory bandwidth - you will get a performance increase in line with the reduction in the code balance.
24	Why is there a peak (MLUPs) at jmax of 1000 in the previous graph?	Until 1000^2 the full data set fits into L3 cache and you are decoupled from main memory. You can see this also in the balance measurements on slide 18 where the measured code balance for problems less equal 1000 is zero as the full data set fits into the large L3 cache
25	Are there more reasons to the flattening of the graph at more cores than the layer condition? No overhead with more cores?	The problem sizes we're dealing with here are such that the typical OpenMP overhead (mostly the barrier at the end of the workshared loop) is entirely negligible. The barrier costs (depending on the number of cores and the implementation in the OMP runtime) a couple of thousand cycles. A full sweep takes much longer than that. The flattening is really just caused by the memory bandwidth saturation here.
26	If you would please talk more about why you did not (or could not) create any test cases that would fit into L1?	In case of the 2d 5pt stencil this is possible, and you can analyze the code and predict the performance if the inner loop is sufficiently long. We don't include such cases in the tutorial because in-cache modeling is much harder than straighforward Roofline analysis, especially when the data is in L2 or L3. Basically you need more advanced models because the basic Roofline assumption that all data transfers

		overlap with execution is not true for in-cache scenarios, especially on Intel CPUs.
27	Where does 3.5 cycles for loading from memory come from (in the SIMD discussion)? Wouldn't this typically be 60+ ns or so?	The 60+ ns are the main memory latency. In our model we assume that latency can be ignored because prefetchers are working perfectly. So the transfer time over the memory bus is determined completely by the bandwidth. The CPU we used for slide 12 has a memory bandwidth of 40 Gbyte/s and a clock speed of 2.2 GHz. One CL transfer (64 byte) thus takes $64/(40*10^9)*2.2*10^9$ cycles = 3.5 cycles. Since the transfers through the cache hierarchy do not overlap, a single core cannot achieve the 40 Gbyte/s because the other transfers also take time, they add to the 3.5 cycles.
28	are there circumstances where is is still beneficial to vectorize even when the data set is much larger than cache and the operations are completely memory bandwidth-bound?	When you really know that a loop is memory bound, the first thing to do is to try and reduce the amount of data transferred to/from the memory, so SIMD is out of the game. There is one exception: Nontemporal stores (a.k.a. streaming stores) only exist in SIMD variants. Thus, a loop with a write-allocated store stream is only a candidate for NT stores if it can also be vectorized. So in this particular case the SIMD vectorization can actually do what you need, i.e., reduce the amount of data loaded from memory.
29	Does likwid-perfctr have any built-in groups for looking at false cacheline sharing?	Yes, on some architectures. E.g., on Ivy Bridge there is the FALSE_SHARE group.
30	How does one control the automatic page migration in the NUMA balancing you mentioned?	To deactivate it under Linux: "echo 0 > /proc/sys/kernel/numa_balancing". Use 1 to reactivate it.
31	If the memory is initialized (not parallely) and if the threads are going to use it locally repeatedly, won't it get cached locally instead of accessing it remotely all the time in ccNUMA. So shouldn't it be just a one time traffic?	Yes. The whole ccNUMA issue is only relevant if the code is memory bound. If you have so much cache reuse as to decouple entirely from the memory bandwidth, ccNUMA effects become marginal.
32	I am heavy on using taskset and assumed that I will get the mem within the task's Numa domain as opposed to using numactl. I assume that is the case, right?	The NUMA issue is independent of which tool you use for pinning your threads. Note, however, that taskset and numactl do not pin individual OpenMP threads; they only restrict the movement of the whole team of threads to the set indicated. Hence, if your team of threads spans multiple ccNUMA domains, taskset and numactl are not sufficient to enforce good locality.
33	Are there any other bottlenecks which arise when you have many more threads/cores/nodes than 8 as you experimented when you initialize parallely?	I assume you are referring to the experiment where we compare parallel placement with round-robin and LD0 placement. As a general rule, data access becomes more "inhomogeneous" as the number of domains increases, so you get larger penalties when not doing optimal placement. If you ask whether it is possible to build very large machines and still scale: There are examples of systems with hundreds of ccNUMA domains, and if you know your first touch rule, these machines scale just fine (e.g., SGI Altix/Ultraviolet).
34	Does likwid-perfctr account for masking of AVX512 instructions?	The events on Intel CPUs do not allow for counting individual SIMD lanes being active. You can only count instructions. This is not a limitation of likwid-perfctr but of the performance counting infrastructure on the CPU. Would be

		nice to have since it would allow proper counting of flops, but we asked Intel about it and they say it's "technically impossible." So there.
35	Would you please describe the performance impact of adding memory channels (ie: going from 4 to 6 per socket), in the cases where cores access memory outside of their local NUMA domain?	If the chip has more memory channels you get more bandwidth. The ccNUMA problem stays the same. Of course, if the memory bandwidth is increased but the inter-domain connections stay the same (bandwidth wise), then the problem with nonlocal access is aggravated.