ABC of modern HPC for LQCD

2. Modern CPU Architecture for LQCD (2)

Issaku Kanamori (R-CCS)

Jan. 28, 2022 @ R-CCS

Outline

1. Brief review of the previous lecture
2. Thread parallelization
3. Process parallelization
4. Summary, or tips for performance tunings

References

- A64FX: https://github.com/fujitsu/A64FX
  especally, A64FX_Microarchitecture_Manual_en_1.6.pdf there
- textbook of computer architecture:

  David A. Patterson and John L. Hennessy, “Computer Organization and Design”
  my version is a Japanese translation of the 5th edition (2014, Elsevier)
  コンピュータの構成と設計第 5 版 上・下 日経 BP 社 （ 2014 ）
  the latest is the 6th edition (2020, Morgan Kaufmann) [Japanese translation: 2021]
- Y. Ajima et al. “The Tofu Interconnect D”, IEEE Cluster 2018, 2018. DOI: 10.1109/CLUSTER.2018.00090

SIMD (last time)

focused on a core
loosely regarded cpu ≃ core

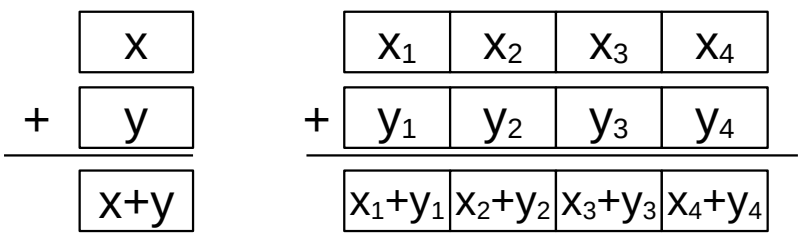
- To increase the performance of one processor
  - increase frequency : stopped at 2-3 GHz heat
  - increase number of cores (~ bundle several small CPUs into 1 processor):

    typical laptop: 2 cores, Fugaku: 48 (+2 or 4) cores, KNL: 64-68 cores

    memory/cache coherence to keep the consistency of the data in memory

    thread parallelization
  - increase the data size processed simultaneously

    Single Instruction Multiple Data (SIMD)



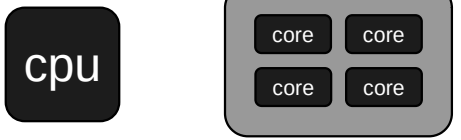
applies the same instruction (+) to several numbers at once

Fugaku: (512bit)
8 double / 16 float / 32 half prec.

Thread/Process Parallelization

cpu = many cores
(+ cache etc.)

- To increase the performance of one processor
  - increase frequency : stopped at 2-3 GHz heat



  - increase number of cores (~ bundle several small CPUs into 1 processor):

    typical laptop: 2 cores, Fugaku: 48 (+2 or 4) cores, KNL: 64-68 cores

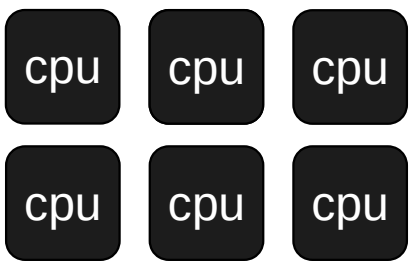
    memory/cache coherence to keep the consistency of the data in memory

    thread parallelization

  - increase the data size processed simultaneously

    Single Instruction Multiple Data (SIMD)

- More CPUs:
  - Tofu network for Fugaku



super computer
= many cpus
(+ memory etc.)

Summary of the last time: 1 or Tips for tuning

- Data access: bandwidth
  - LQCD is memory bandwidth limited application
    B/F of Dirac op. mult ~ 1 >> typical CPUs
    (B/F of Fugaku ~ 0.3 is rather large nowadays)
  - Once data is loaded to the cache, reuse it as much as possible
    loop tiling, cache blocking
- Data access: latency
  - Main memory is far away: O(100) cycles
  - data prefetch

Summary of the last time: 2 or Tips for tuning

- SIMD
  - Without SIMD, the theoretical peak performance would be 1/8 (double prec.) or 1/16 (single prec.)
  - use SIMD friendly data layout: array of structure of array
  - single perc. floating operation is x2 faster (half prec. x4 faster) than double precision. Use mixed prec. algorithm

Summary of the last time: 3 or Tips for tuning

- Out-of-Order, Pipeline etc.
  - Each instruction has latency, ~10 cycles on Fugaku (larger than intel)
  - Execution order can be reordered to reduce waiting time of data
  - Loop fission (or fusion), loop unrolling for pipeline execution

Comment on register spill

Not much difficult to find the spill on Fugaku

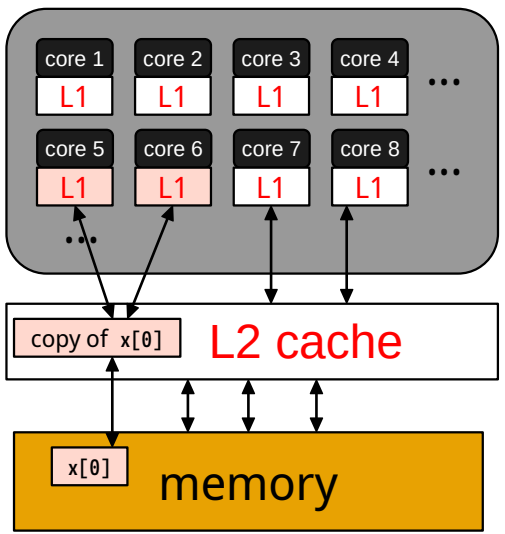
- assemble code generated by -S option tells us register spill

```
.loc    3 130 10 is_stmt 0      // ./inline/vsimd_float-inc.h:130:10
add     x17, sp, #160           // =160
str     z3, [x17]               // 16-byte Folded Spill
```

- `grep “Spill” xxx.s |wc` counts the number of spills

Many Cores: core memory group

- consistency of the contents on cache (cache coherency)
  difficult with many cores



suppose....
core 5 and core 6 have the same data on L1: `x[0] .... x[99]`

- core 5 write `x[0]` (to the L1)
- L1 of core 6 must be updated before using `x[0]`

cache line that contains `x[0]` will be updated

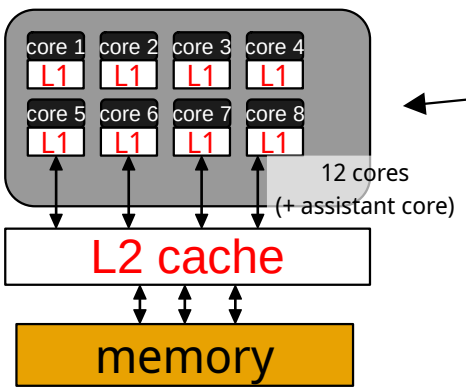
the mechanism to keep the coherency is beyond this lecture
(and I am not familiar with this)

mechanism on Fugaku: MESI Protocol

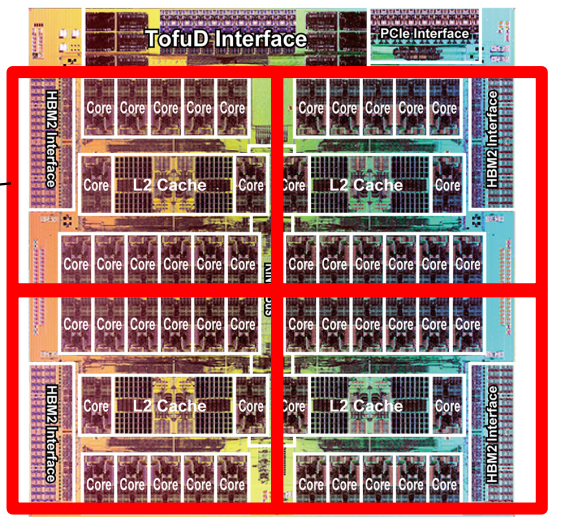
cf. KNL: 2 cores share 1 L2 cache

Reducing number of cores: core memory group

- 48 cores (Fugaku) are divided into 4 gropes (core memory group, CMG)
- each group has its own memory [non uniform memory access, NUMA]
  L2 coherence btw CMGs: cache coherence NUMA (ccNUMA)
- memory access to the other CMG has more latency
- recommend: 1 MPI proc. / CMG



4 CMGs on A64FX



https://www.r-ccs.riken.jp/fugaku/system/ (photo by Fujitsu)

Need enough cores to saturate the memory BW

- memory band width of single core is small
- 4 cores are needed to saturate BW in CMG
- ≥ 12 cores requires memory access over CMGs

plot From:
C. Alappat, J. Laukemann, T. Gruber, G. Hager; G. Wellein, N. Meyer, T. Wettig
arXiv:2009.13903 [cs.PF]
2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)

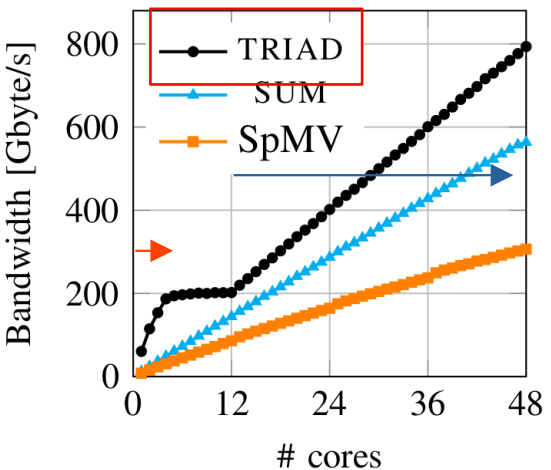


Fig. 1. Scaling of STREAM TRIAD, SUM reduction, and SpMV in CRS format with the HPCG matrix using gcc version 10.1.1. The working set size is 4 GB for TRIAD and SUM; for HPCG the dimension is 128^3.

Thread

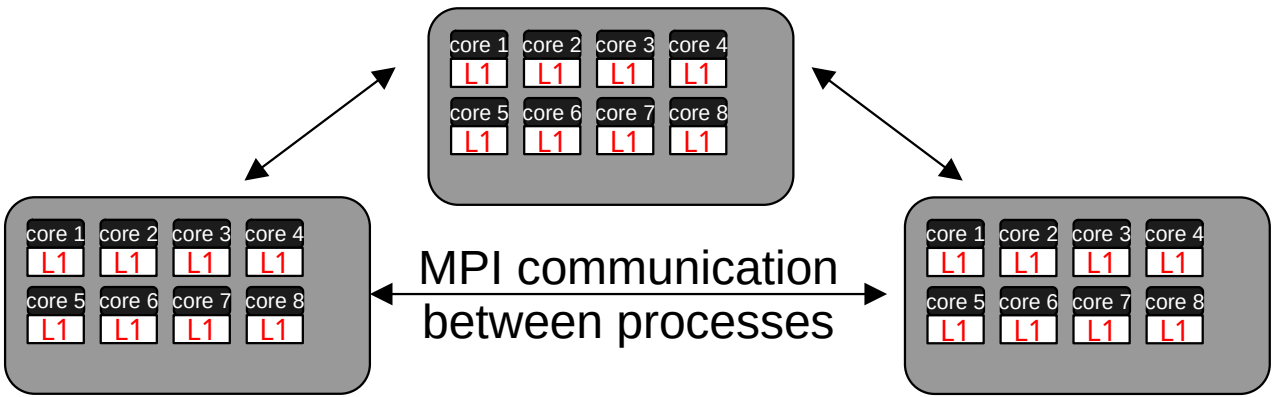
- thread $\simeq$ (program running on a) core
- no hyper-threading on Fugaku

  KNL: up to 4 hyper-threading, 4 x 32 registers etc. but only 2 arithmetic logic units (ALU)
- Shared memory: each thread shares the global memory
  cf. MPI process: each process has independent memory
- Synchronization between threads takes some time: O(100 cycles)
  fast hardware barrier in CMG is available with Fujitsu OMP

```
#pragma omp barrier
```

Hybrid Parallelization: OpenMP + MPI

- each process has several threads (12 threads for 1 MPI proc./CMG)
- 2 layers: thread level (=core) and process level
- which thread from 12 threads calls MPI ?



which thread calls MPI?

- case 1: only one thread (MPI_THREAD_SINGLE)
- case 2: only the thread that called MPI_Init_thread (MPI_THREAD_FUNNELED)
- case 3: only on thread at one time (MPI_THREAD_SERIALIZED)
- case 4: any thread any time (MPI_THREAD_MULTIPLE) not supported on Fugaku

to use communication in thread-parallel, need to call low level interface called uTofu QCD Wide Simd Library (QWS) and LDDHMC uses uTofu

Acceleration of the communication

- communication: copy data from memory to the RDMA engine

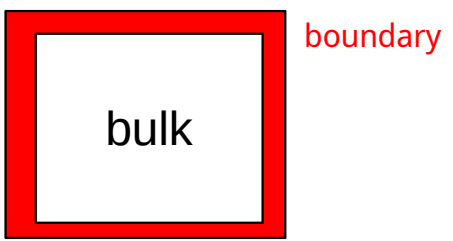
  something like network card

  RDMA: remote direct memory access

- Who/How to takes care the copy: implementation dependent
  - Fugaku: assistant cores can help (Fujitsu extension of MPI)
  - KNL (Oakforest-PACS): 1 thread should take care of it

    frequently calling MPI_Test or immediately calling MPI_Wait without bulk computation

1. pack boundary data to buffer
2. send the boundary data
3. computation in the bulk (communication is overlapped)
4. wait the boundary data arrives
5. computation in the boudary area



Network of Fugaku: Tofu

- 6 dimensional mesh-torus

  Tofu = TOrus FUsion

  6 dim = (3 large dim) x (3 compact dim)

  $\simeq (T_2 \times I) \times (Z_2 \times Z_3 \times Z_2)$ closed (torus)

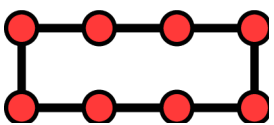
  = Tofu

  $(X \times Y \times Z) \times (A \times B \times C)$

  $= (24 \times 23 \times 24) \times (2 \times 3 \times 2)$

- (1 large dim) x (1 compact dim) can make a close loop

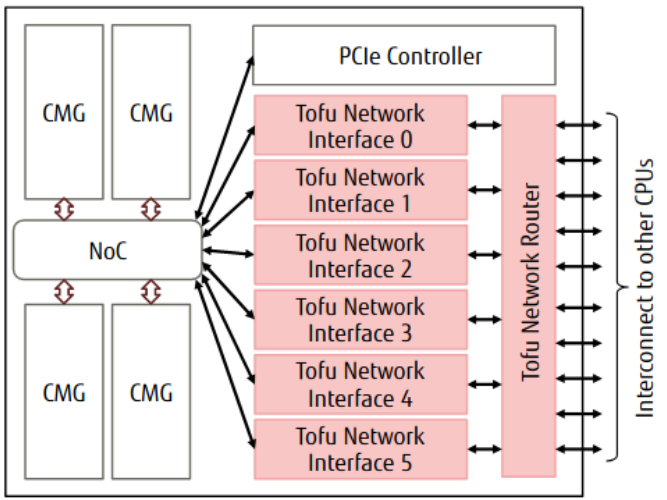
  3 closed loop : 3-dim torus network



4 (large) x 2 (compact)

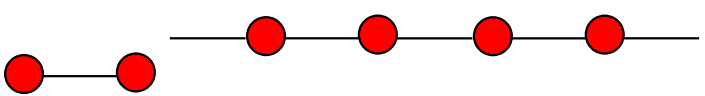
TofuD interconnect

- 6 Tofu Network Interaces (TNIs)
  - Bandwidth 6.8 GB/s each
    (measured 6.35 GB/s)
  - Latency 0.49-0.54 μs (measured)
    $\simeq$ 1000 cycles
  - uTofu interface allows user to specify which TNI to use



taken from
https://www.fujitsu.com/global/documents/about/resources/publications/technicalreview/2020-03/article03.pdf

- connecting with 10 neigbors
  2 neighbors in each direction
  except for Z_2



Tofu Barrier

- barrier synchronization of processes O(10000) cycles
- Allreduce (e.g. global sum over processes) at the same
  up to 3 flaoting point numbers
  - reduces the cost for inner product <x|y> in iterative solver
  - out come of the co-design with LQCD

Summary, or Tips for tunig

- NUMA structure [A64FX]
  - 48 cores = 4 x 12 cores
  - each CMG has own memory + L2 cache
- thread parallelization [multi cores]
  - cache coherence is needed
  - needs several threads to saturate memory bandwidth
  - memory access over CMGs takes more time
    (first touch --- a major technique for intel machines)
  - synchronization: O(100) cycles or more

Summary, or tips for tuning

- process parallelization [multi CPUs, multi CMGs] O(10000) cycles
- MPI_THREAD_MULTIPLE is not supported
  need to use uTofu for thread parallel communication
- 6-dim mesh torus network
  rankmap to match 4-dim LQCD lattice to 6-dim node network

What’s next?

- QCD Wide SIMD Library (QWS)
  algorithm (+ implementation) cf. K.-I. Ishikawa et al, 2109.10687
- FLOP counting of Dirac operators
  Wilson and Clover
- an Example of low level tuning