

Tutorial / Hands-on

Scientific Benchmarking

ACM ASEAN School on HPC and AI
12. Mar. 2025

Jens Domke Dr. rer. nat. < jens.domke@riken.jp >
Supercomputing Performance Research Team, RIKEN R-CCS, Kobe, JP

RIKEN R-CCS Computer simulations create the future

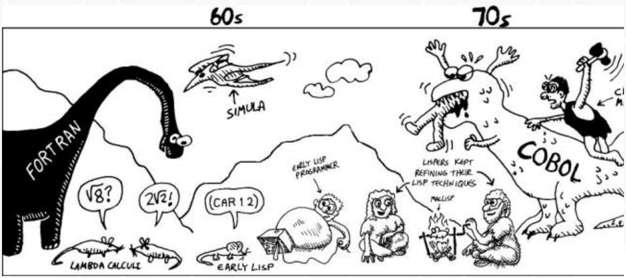
Let’s recap Scientific Benchmarking

Computer simulations create the future

- **Objective:**
  - **Your boss:** Hey, what’s the **peak triad** bandwidth of the new FX1000 A64FX CPU?
  - **Smart you:** *(you don’t fool me twice ☺)* I’ll check this STREAM Triad benchmark which I found online!

- **Step 1** – getting code from your friendly benchmark dealer

```
cd ~
pjsub --interact -x PJM_LLIO_GFSCACHE=/vol0006:/vol0004 -g "hp240465" --mpi "max-proc-per-node=1" -L "rscgrp=excl_hp240465_2503-1" -L "elapse=02:00:00" --sparam "wait-time=600" --no-check-directory
cp -r /vol0601/d*/hp240465/Mat*/AI*HPC/SB .
cd SB/ && ls
less stream.F
```



- **Step 2** – WTF!? is Fortran77

- **Step 3** – test

```
make -B
./stream
```

Reporting Result – Attempt 1

Computer simulations create the future

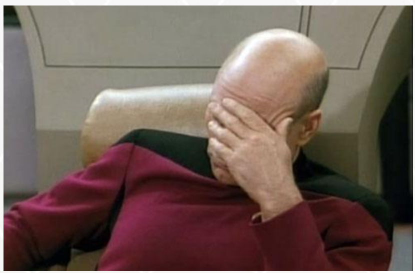
- First run

Four kernels, separately timed:

```
Copy:   C[i] = A[i];   i=1..N
Scale:  B[i] = scalar * C[i]; i=1..N
Add:    C[i] = A[i] + B[i]; i=1..N
Triad   A[i] = B[i] + scalar*C[i]; i=1..N
```

```
-------------------------------------
STREAMy Version $ Revision: 3.14
-------------------------------------
Array size =          64
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
---------------------------------------------
---------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
---------------------------------------------------
Function      Rate (MB/s)  Avg time   Min time  Max time
Copy:         Infinity      0.0000     0.0000     0.0000
Scale:        Infinity      0.0000     0.0000     0.0000
Add:          Infinity      0.0000     0.0000     0.0000
Triad:        Infinity      0.0000     0.0000     0.0000
---------------------------------------------------
Solution Validates!
---------------------------------------------------
```

- You vs. your boss



What went wrong here?

Computer simulations create the future

- First run

```
-------------------------------------
STREAMy Version $ Revision: 3.14
-------------------------------------
Array size =          64
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
---------------------------------------------
-------------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
-------------------------------------------------------
Function      Rate (MB/s)  Avg time    Min time    Max time
Copy:         Infinity       0.0000      0.0000      0.0000
Scale:        Infinity       0.0000      0.0000      0.0000
Add:          Infinity       0.0000      0.0000      0.0000
Triad:        Infinity       0.0000      0.0000      0.0000
-------------------------------------------------------
Solution Validates!
-------------------------------------------------------
```

Four kernels, separately timed:

```
Copy:  C[i] = A[i];             i=1..N
Scale: B[i] = scalar * C[i];    i=1..N
Add:   C[i] = A[i] + B[i];      i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

- Four obvious mistakes
  - Array size?
  - Rate calculation?
  - Incorrect results?
  - Timer granularity?
- Any non-obvious mistakes?
  - …

What went wrong here?

- **Step 4** – Looks for anything related to “array size” or “64”

```
vim stream.F
```

- **Step 5** – How big should the **array size** be?
  - 128
  - 1.000.000
  - 12*1024*1024/3/8
  - 32*10^9/8
  - (other)

- **Step 6** – Recompile & run

```
make -B
./stream
```

Reporting Result – Attempt 2

Computer simulations create the future

- 1GB memory per run

Four kernels, separately timed:

```
Copy:  C[i] = A[i];            i=1..N
Scale: B[i] = scalar * C[i];   i=1..N
Add:   C[i] = A[i] + B[i];     i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

```
-------------------------------------------------------------
STREAMy Version $ Revision: 3.14
-------------------------------------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
----------------------------------------------
-------------------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
-------------------------------------------------------------
Function      Rate (MB/s)   Avg time    Min time   Max time
Copy:        13112.1182      0.0546      0.0546      0.0546
Scale:       13073.7498      0.0548      0.0548      0.0548
Add:         14174.4314      0.0758      0.0758      0.0758
Triad:       14191.4937      0.0757      0.0757      0.0757
-------------------------------------------------------------
Solution Validates!
-------------------------------------------------------------
```

- You vs. your boss

14191 MB/s
=14 GB/s ☺

How much of peak?

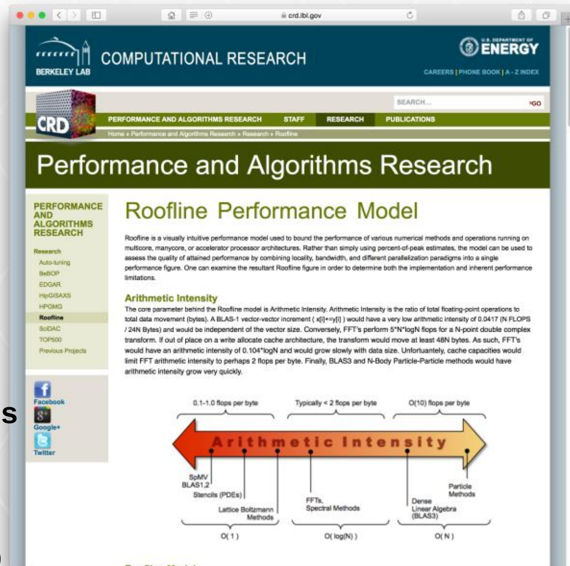
Where on the roofline?

Roof? Line? What?

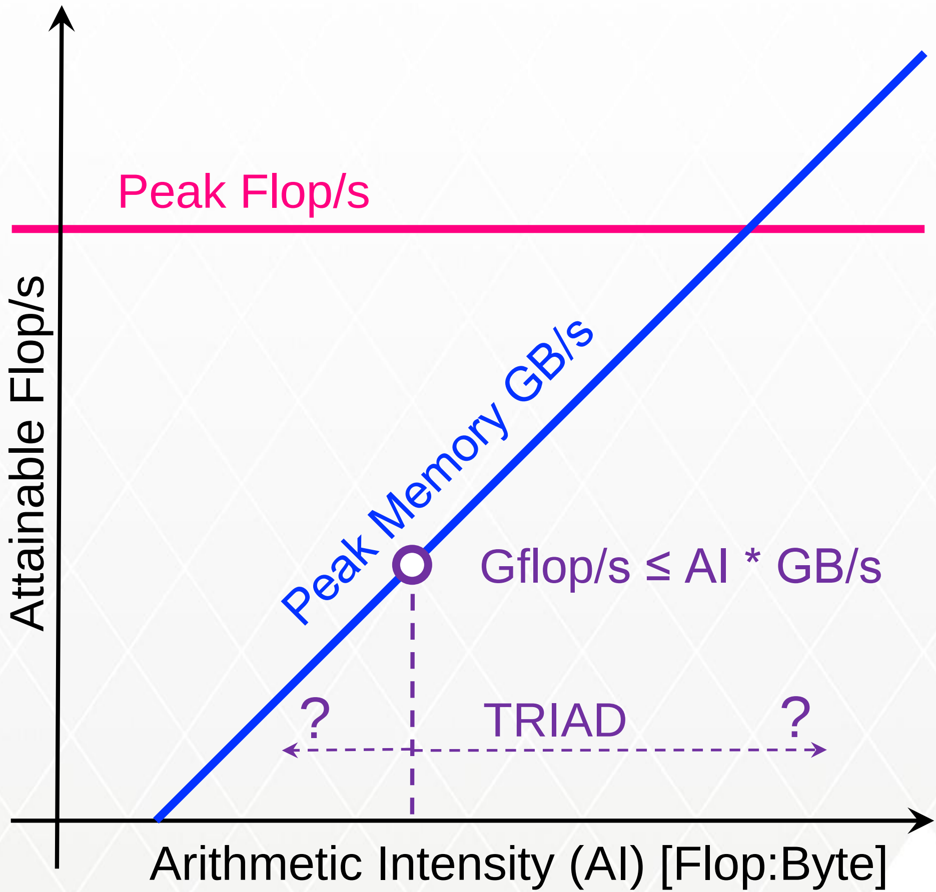
Little excursion – Roofline Model

Computer simulations create the future

- The **Roofline Model is a throughput-oriented performance model**…
  - Tracks rates not times
  - Augmented with Little's Law (concurrency = latency*bandwidth)
  - Independent of ISA and architecture (applies to CPUs, GPUs, Google TPUs, etc…)
- **Three Components**:
  - **Machine Characterization** (realistic perf. potential of the system)
  - **Application Execution Monitoring**
  - **Theoretical Application Bounds** (how well could my app perform with perfect compilers, caches, overlap, …)



Roofline Sketch



Reporting Result – Attempt 3

Four kernels, separately timed:

```
Copy:  C[i] = A[i];           i=1..N
Scale: B[i] = scalar * C[i];  i=1..N
Add:   C[i] = A[i] + B[i];    i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

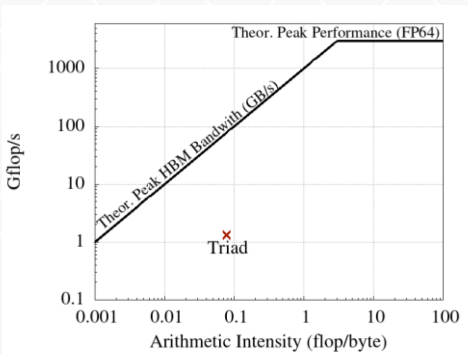
- **Step 7** – Arithm. Intensity for STREAM Triad
  - ??? floating-point operations per index `i`
  - ??? bytes per index `i`

  ➔ AI of Triad == ???

- **Step 8** – Plot roofline (local system; need gnuplot)

```
scp <fugaku>:SB/roofline.gp .
scp <fugaku>:SB/fugaku.data .
# fill out fugaku.data & roofline.gp
gnuplot roof*.gp && showimage2 roof*.png
```

☹☹☹



What went wrong here?

- **Step 9** – Brainstorm other simple explanations?
  - 1024 GB/s peak / 14 GB/s achieved ~= 70
  - ➔ We have 48 cores which is close to 70 ☺
  - ➔ Any parallelization issues???

- **Step 10** – Look for anything related to **MPI or OpenMP**

```
vim Makefile
```

- **Step 11** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 ./stream
```

Reporting Result – Attempt 4

Computer simulations create the future

- OpenMP fixed for run

Four kernels, separately timed:

```
Copy:  C[i] = A[i];   i=1..N
Scale: B[i] = scalar * C[i]; i=1..N
Add:   C[i] = A[i] + B[i]; i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

```
STREAMy Version $ Revision: 3.14
---------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
----------------------------------------------
Number of Threads =           48
----------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
----------------------------------------------------
Function      Rate (MB/s)  Avg time   Min time  Max time
Copy:       55585.5835     0.0129     0.0129     0.0129
Scale:      55696.9485     0.0129     0.0129     0.0129
Add:        63920.7386     0.0168     0.0168     0.0168
Triad:      63890.8136     0.0168     0.0168     0.0168
----------------------------------------------------
Solution Validates!
```

- You vs. your boss

  64 GB/s ☹
  (only 5x faster with
  48 cores)

  still not great ☹

What went wrong here?

Computer simulations create the future

- **Step 12** – Brainstorm other simple explanations?
  - Array length OK; threads OK; …
  - ➔ Reading more Fugaku manuals

- **Step 13** – Let's try **another compiler** (e.g. Fujitsu's fcc/frt)

```
vim Makefile
```

- **Step 14** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 ./stream
```

Reporting Result – Attempt 5

Computer simulations create the future

- Compiler change for run

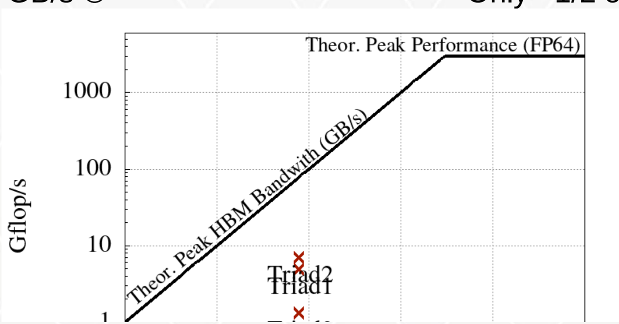
Four kernels, separately timed:

```
Copy:  C[i] = A[i];              i=1..N
Scale: B[i] = scalar * C[i];     i=1..N
Add:   C[i] = A[i] + B[i];       i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

```
STREAMy Version $ Revision: 3.14
-------------------------------------------------------------
Array size =    44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
----------------------------------------------
Number of Threads =  48
----------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
----------------------------------------------------
Function      Rate (MB/s)   Avg time    Min time   Max time
Copy:        87776.6323      0.0082      0.0082      0.0082
Scale:       86799.6446      0.0082      0.0082      0.0082
Add:         90511.8789      0.0119      0.0119      0.0119
Triad:       91071.9613      0.0118      0.0118      0.0118
----------------------------------------------------
Solution Validates!
```

- You vs. your boss

91 GB/s ☹ Only ~1/2 of 1 HBM



What went wrong here?

- **Step 15** – Brainstorm other simple explanations?
  - Array length OK; threads OK; compiler changed; …
  - ➔ Recall boss: Only ~1/2 of 1 HBM
  - ➔ Maybe we only use 1 HBM module out of 4 ???
  - ➔ Recall OpenMP lecture

- **Step 16** – Let's check **first-touch** for array initialization

```
vim stream.F
```

- **Step 17** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 ./stream
```

Reporting Result – Attempt 6

Computer simulations create the future

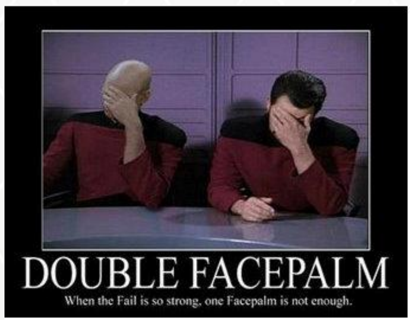
- First-touch memory run

Four kernels, separately timed:

```
Copy:  C[i] = A[i];  i=1..N
Scale: B[i] = scalar * C[i]; i=1..N
Add:   C[i] = A[i] + B[i]; i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

```
STREAMy Version $ Revision: 3.14
---------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
---------------------------------------------
Number of Threads =  48
---------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
---------------------------------------------------
Function      Rate (MB/s)  Avg time   Min time  Max time
Copy:      **********     0.0013     0.0013     0.0013
Scale:     **********     0.0013     0.0013     0.0013
Add:       **********     0.0017     0.0017     0.0017
Triad:     **********     0.0017     0.0017     0.0017
---------------------------------------------------
Solution Validates!
```

- You vs. your boss



What went wrong here?

Computer simulations create the future

- **Step 18** – What now?
  - Array length OK; threads OK; compiler changed; first-touch fixed; …

➔ Not enough iterations?

➔ Cosmic rays?

➔ Wrong print format?

➔ (other)?

- **Step 19** – Let’s check all **(iterations counter; prints; …)**

```
vim stream.F
```

- **Step 20** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 ./stream
```

Reporting Result – Attempt 7

- Outdated BM fixed for run

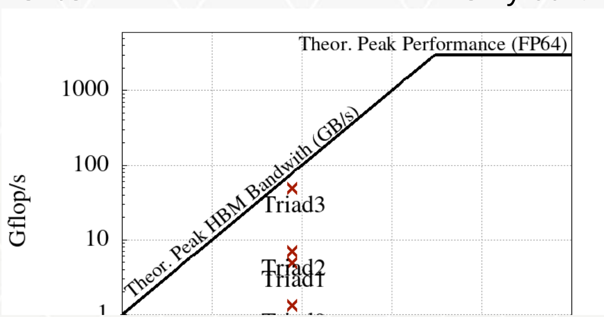
Four kernels, separately timed:

```
Copy:  C[i] = A[i];  i=1..N
Scale: B[i] = scalar * C[i]; i=1..N
Add:   C[i] = A[i] + B[i]; i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

```
STREAMy Version $ Revision: 3.14
-------------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
----------------------------------------------
Number of Threads =  48
----------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
----------------------------------------------------
Function      Rate (MB/s)  Avg time   Min time  Max time
Copy:        552826.3132       0.0013       0.0013       0.0013
Scale:       549286.4447       0.0013       0.0013       0.0013
Add:         622130.0677       0.0017       0.0017       0.0017
Triad:       631552.3153       0.0017       0.0017       0.0017
----------------------------------------------------
Solution Validates!
```

- You vs. your boss

631 GB/s Only 60% of peak ☹



What went wrong here?

Computer simulations create the future

- **Step 21** – Brainstorm other simple explanations?
  - Array length OK; threads OK; compiler changed; first-touch fixed; fixed iterations+print; …

  ➔ Different compiler flags?

  ➔ Different runtime parameters?

  ➔ Problems with Fujitsu's hugepages memory allocations?

  ➔ Reading MORE of Fugaku manual…

- **Step 22** – Let's try fixing the **page allocation** scheme

```
#change XOS_MMM_L_PAGING_POLICY=…
```

- **Step 23** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 \
XOS_MMM_L_PAGING_POLICY=prepage:prepage:prepage \
./stream
```

Reporting Result – Attempt 8

- Try page allocation for run

```
STREAMy Version $ Revision: 3.14 $
-------------------------------------------------------------
Array size =    44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
----------------------------------------------
Number of Threads =  48
----------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
----------------------------------------------------
Function      Rate (MB/s)  Avg time   Min time  Max time
Copy:         87263.8408       0.0082        0.0082        0.0083
Scale:        87756.1075       0.0082        0.0082        0.0082
Add:          90502.7845       0.0119        0.0119        0.0120
Triad:        91812.7612       0.0117        0.0117        0.0118
----------------------------------------------------
Solution Validates!
```

Four kernels, separately timed:

```
Copy:  C[i] = A[i];            i=1..N
Scale: B[i] = scalar * C[i];   i=1..N
Add:   C[i] = A[i] + B[i];     i=1..N
Triad  A[i] = B[i] + scalar*C[i]; i=1..N
```

- You vs. your boss

  back to 91 GB/s?
  WHY???
  (looks same as Attempt 5
  ➔ “broke” first-touch?)

What went wrong here?

Computer simulations create the future

- **Step 24** – Brainstorm other simple explanations?
  - Array length OK; threads OK; compiler changed; first-touch fixed; fixed iterations+print; on-demand pages; …

  ➔ Different compiler flags?

  ➔ ~~Different runtime parameters?~~ back to on-demand

  ➔ ~~Problems with Fujitsu's hugepages memory allocations?~~

  ➔ Reading MORE of Fugaku manual…

- **Step 25** – Let's try fixing the page allocation scheme

```
vim Makefile
#add -Kprefetch_sequential=soft \
     -Kprefetch_line=??? \
     -Kzfill=???
```

Which numbers?
➔ RNG: 12 & 14

- **Step 26** – Recompile & run

```
make -B
OMP_NUM_THREADS=48 \
XOS_MMM_L_PAGING_POLICY=demand:demand:demand \
./stream
```

Reporting Result – Attempt 9

Computer simulations create the future

- Outdated BM fixed for run

Four kernels, separately timed:

```
Copy:   C[i] = A[i];   i=1..N
Scale:  B[i] = scalar * C[i];  i=1..N
Add:    C[i] = A[i] + B[i];  i=1..N
Triad   A[i] = B[i] + scalar*C[i];  i=1..N
```

```
STREAMy Version $ Revision: 3.14 $
-------------------------------------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
---------------------------------------------
Number of Threads =  48
-------------------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
-------------------------------------------------------------
Function      Rate (MB/s)   Avg time    Min time   Max time
Copy:        576276.3353       0.0012       0.0012       0.0013
Scale:       627329.6504       0.0012       0.0011       0.0012
Add:         726973.2946       0.0015       0.0015       0.0015
Triad:       738537.1532       0.0015       0.0015       0.0015
-------------------------------------------------------------
Solution Validates!
```

- You vs. your boss

Nice 739 GB/s ☺

What went wrong here?

Computer simulations create the future

- **Step 27** – Brainstorm other simple explanations?
  - Array length OK; threads OK; compiler changed; first-touch fixed; fixed iterations+print; on-demand pages; memory flags

  ➔ Do we know the interaction between zfill and prefetch flags?

  ➔ Yes → use Fractional Factorial
No → maybe (pruned) Full Factorial Experiment Design

- **Step 28** – Let's try **pruned full factorial** (`frt` manpage: both flags can be 1≤n≤100 → 100x100 runs is crazy slow)

```
vim Makefile
```

```
#add -Kprefetch_sequential=soft \
     -Kprefetch_line=$(PREF) \
     -Kzfill=$(ZFILL)
```

```
vim sweep.sh
# try from 8 to 18 (which are around 12/14)
```

- **Step 29** – Sweeping compile+run experiment

```
bash ./sweep.sh
```

Reporting Result – Attempt 10

- Sweeping for best run

```
14 18 Triad: 595556.6729 0.0018 0.
16 8 Triad: 830462.7624 0.0013 0.0
16 10 Triad: 840067.0696 0.0013 0.
16 12 Triad: 843370.7042 0.0013 0.
16 14 Triad: 731341.2732 0.0015 0.0015 0.0015
16 16 Triad: 486297.3286 0.0022 0.0022 0.0022
16 18 Triad: 519446.3161 0.0021 0.0021 0.0021
18 8 Triad: 829086.8115 0.0013 0.0013 0.0013
```

Four kernels, separately timed:

```
Copy:   C[i] = A[i];   i=1..N
Scale:  B[i] = scalar * C[i];  i=1..N
Add:    C[i] = A[i] + B[i];  i=1..N
Triad   A[i] = B[i] + scalar*C[i];  i=1..N
```

- Getting full output

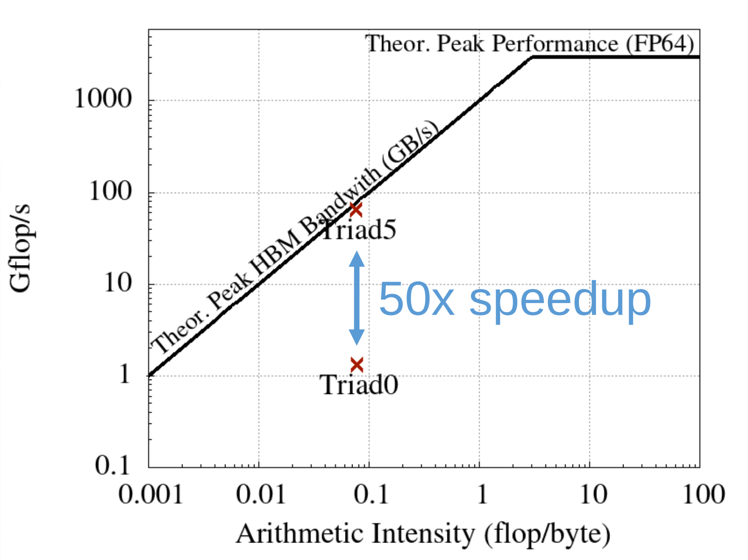
```
   ZFILL=16 PREF=12 make -B
   OMP_NUM_THREADS=48 \
XOS_MMM_L_PAGING_POLICY=demand:demand:demand \
./stream
```

```
STREAMy Version $ Revision: 3.14 $
-------------------------------------------------------------
Array size =   44739242
The total memory requirement is 1023 MB
--
The *best* time for each test is used
*EXCLUDING* the first and last iterations
-------------------------------------------------------------
Number of Threads =  48
-------------------------------------------------------------
Your clock granularity/precision appears to be      1 microseconds
-------------------------------------------------------------
Function     Rate (MB/s)  Avg time   Min time  Max time
Copy:       774612.9275       0.0009        0.0009        0.0009
Scale:      771428.4961       0.0009        0.0009        0.0009
Add:        838190.8729       0.0013        0.0013        0.0013
Triad:      843528.6683       0.0013        0.0013        0.0013
-------------------------------------------------------------
Solution Validates!
```

Reporting Result – Attempt 10

Computer simulations create the future

- **Step 30** – Final roofline plot



- You vs. your boss

845 GB/s ☺ ☺ ☺