A proposal for "New" benchmark: HPL with time constraint

Jun Makino

Group seminar Dec 16 2015

Before the main topic...

How I Learned to Stop Worrying and...

Practical points

- **I will officially move to Kobe University as of March 1st, 2016. (Department of Planetology, Graduate School of Science)**
- **However, I'll remain in AICS, RIKEN as a "part-time" TL and deputy project leader, and there will be no change in position of anybody in AICS (including JRA and Special/Foreign Postdoctoral Researcher)**

Bottom line of my talk today

- **What "new" benchmark?**
  - **HPL with time limit of 10, 300, and 10,000 sec.**
- **Why new benchmark?**
  - **Measure three important aspects of performance**
  - **Guide the developers (and buyers) of large-scale HPC machines**

Structure of Talk

- Present status and problems of HPL
- How about HPCG?
- Performance characteristics of HPL
- What do we want to know?
- Problem size, efficiency and execution time
- “New” benchmark

What is HPL?

- High Performance Linpack: used for more than two decades for the Top 500 ranking
- Specification: Solve a dense matrix using the direct method. Optimizations which reduced the operation count are prohibited (e.g. Strassen's algorithm for matrix multiplication)
- Problem size is not specified (in old "LINPACK Benchmark", problem size was $N = 100$ and 1000).

Advantages of HPL

- None of the source program, algorithm, or problem size are specified. Thus, can be used for new hardware/new algorithms without any change in the benchmark rules
- Can compare machines of quite different scales

These advantages are quite important and these are reason why HPL has been used for long time.
Almost disappeared benchmarks: SLALOM, NAS parallel, SpecHPC ....

Disadvantages of HPL

- Due to the great advance in optimization and introduction of new algorithm (most importantly Gustavson's recursive blocking) and the lack of problem size limit, required bandwidth (memory and network) is minimized
- Therefore, machines with very weak memory and network can still achieve good performance if the problem size is large enough.
- Measurements on big machines take very long time (30h on K computer).

Discrepancy with the performance of real applications is rather serious. HPL can achieve 50-80%, while it is not unusual to see "my code" running with 5% efficiency

Dongarra's "answer"

"HPCG"

- Benchmark for multicolor CG iteration for FEM.
- Actual grid is regular, but program rewrite to eliminate indirect access is prohibited.

K computer: 5%SX-ACE: around 10%.

Is HPCG the answer?

It does have several important advantages and solved major problems of HPL.

- **Most important is the performance of memory, in particular the gather operation. Not determined by the arithmetic performance only.**
- **Execution time is not so long.**

Problems of HPCG

- Unlike HPL, the algorithm (and therefor the memory access pattern) is fixed. No room for taking into account the future evolution of architecture and algorithms.
- Therefore, if real applications evolve to reduce the memory access, HPCG will be left behind.
- In fact, already it is.
- Theoretically, the indirect access can be avoided without violating the rule, by some rewriting of the code and compiler hack. Once this is done, this benchmark will be useless. (Probably...)

Optimization in Gray zone

- While retaining the indirect access in the source program, it is possible to rewrite the data and loop structures so that all memory access are actually contiguous.
- After doing so, the following "optimization" is theoretically possible:
  - For a vectorizable loop with indirect access, the compiler generates the code to test if the array index is actually contiguous or linear.
  - If the array index is contiguous or linear, the code with no indirect access is executed.
- It is also possible to handle this with the special implementation of gather load instruction. If the addresses are actually contiguous, switch to normal load.

So

- HPCG considered harmful.
- On the other hand, currently HPL is problematic.

What we need?

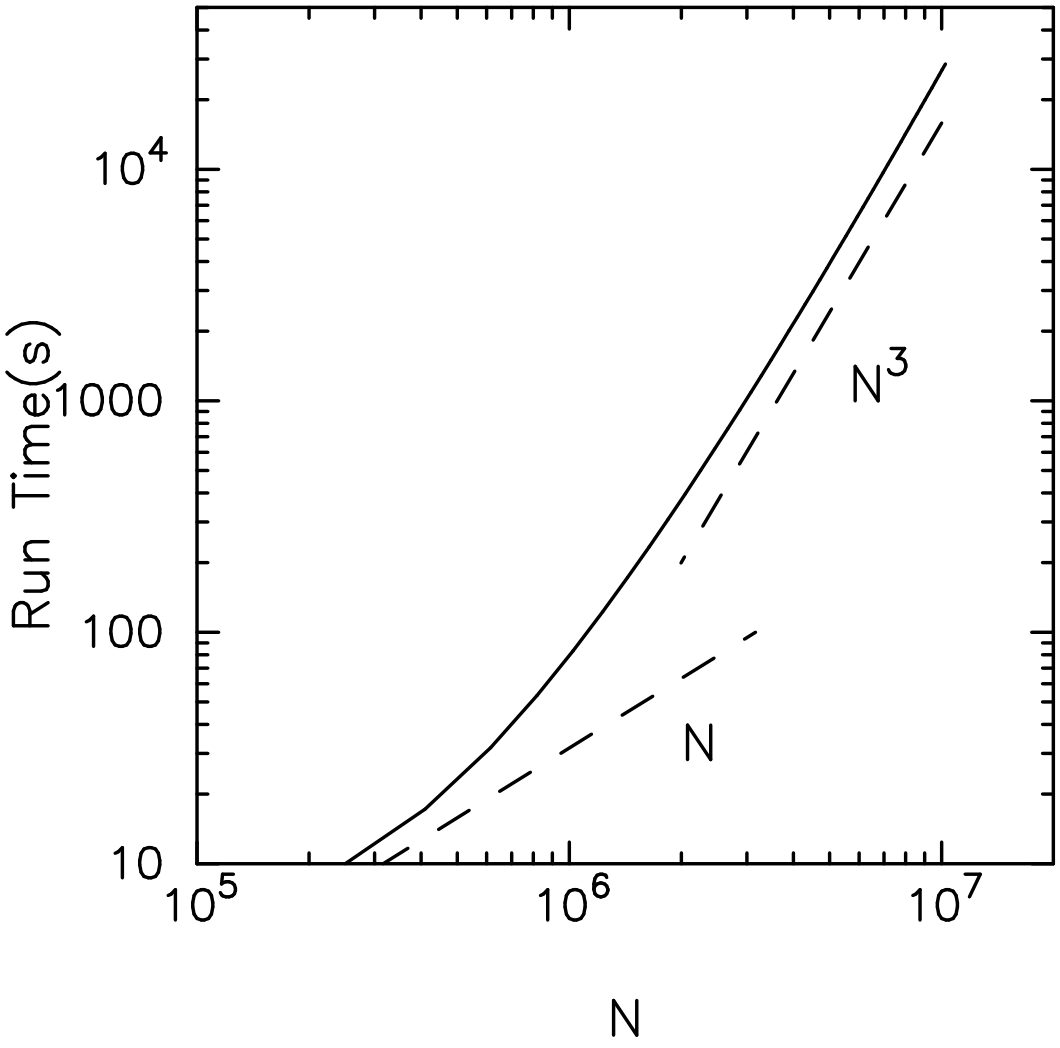
- HPL measures the arithmetic performance, and to some extent network bandwidth.
- Network latency is not measured at all.
- Execution time is critical problem
- My opinion is that one should not put too much stress on memory performance, in particular that of gather/scatter.

So what should we do?

Our idea

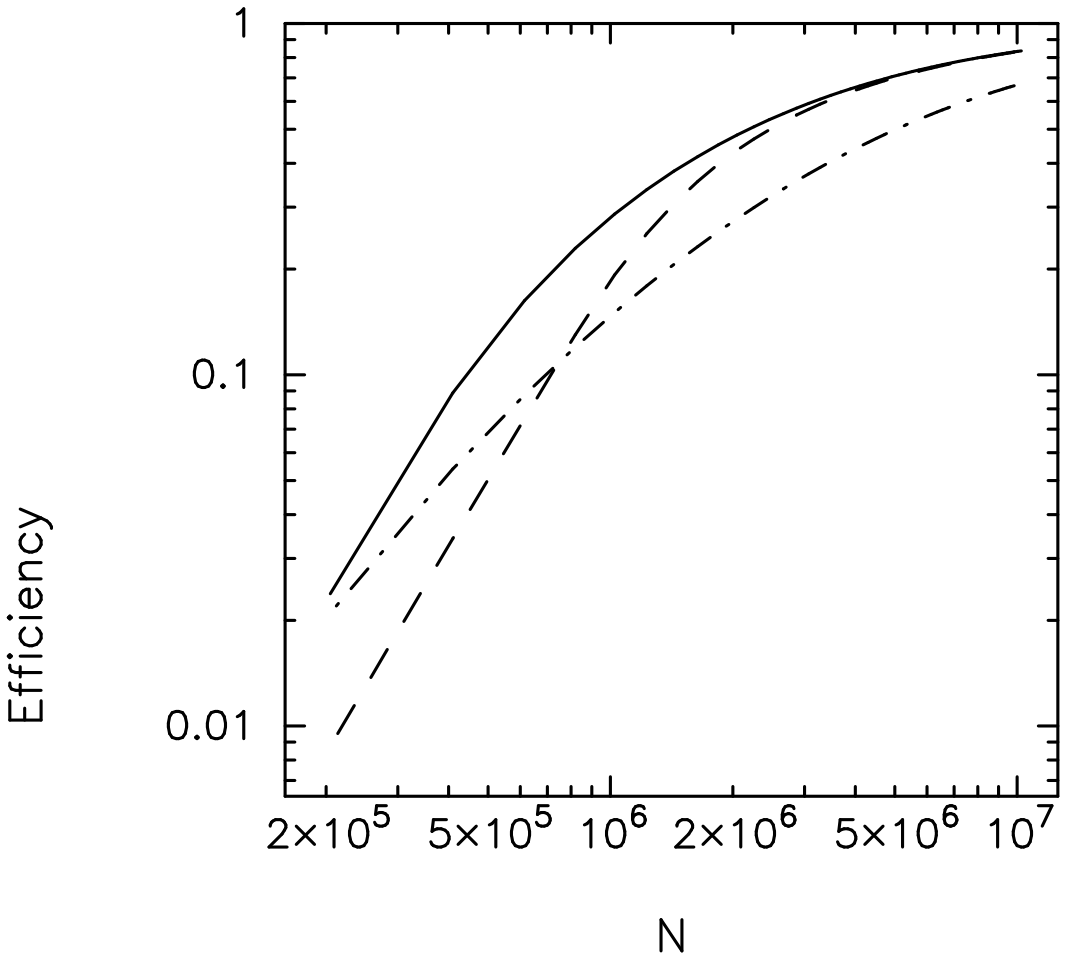
- If HPL takes too much time, should we jut put the time limit?
- To measure latency, can we use execution with some fairly short time limit?

How the "HPL with time limit" would look like?



A hypothetical machine (node performance 3TF 10K nodes) Performance model I made (improved version of what is on the netlib HPL page) Network: 30Gbps Communication latency: 5μs.

Efficiency of a hypothetical machine



Latency limited for small size
Network limited for intermediate size
Approaches to peak for large size
Dashed: large latency
Dot-dashed: low bandwidth

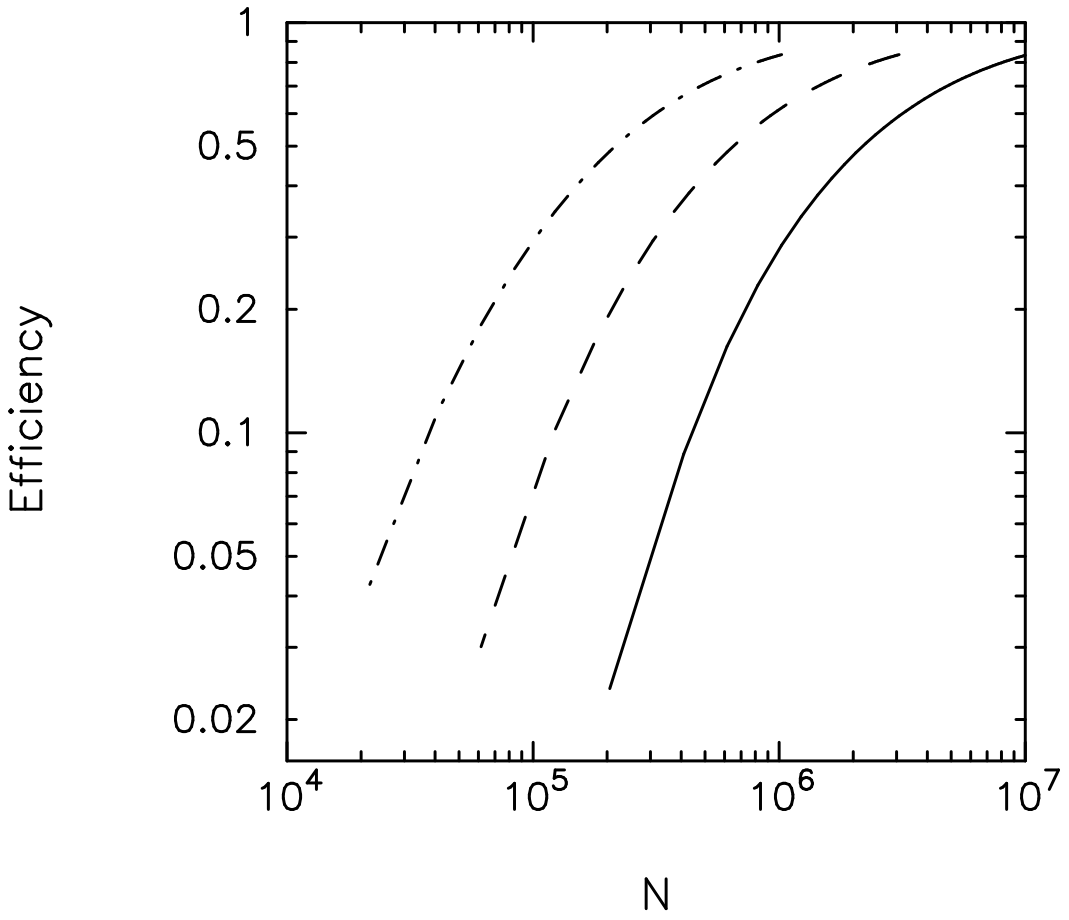
What does this result tells us?

- For many machines, HPL results for three different sizes would be enough to tell how they behave.
- Meaninglessly high efficiency is achieved if there is no limit in size or time.

Problem:

The best sizes to measure would depend on the machine size.

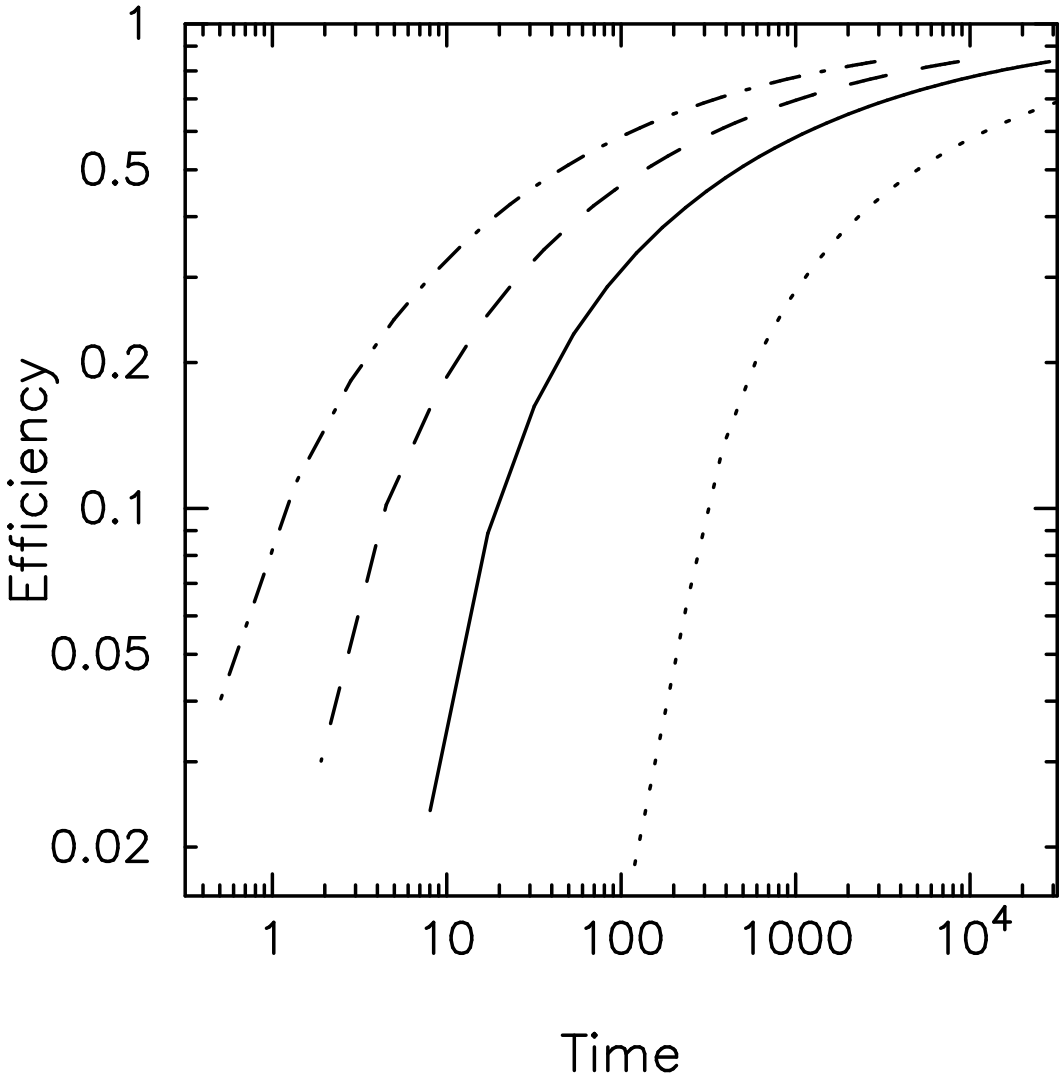
Efficiency and machine size



Machines of size 1/10, 1/100 We can see that the critical problem size depends on the machine size. Not quite ideal.

So what should we do?

Execution time and efficiency



Much better than using the same size.
Of course, for short time the efficiency of large systems are low, but isn't it the problem of the big machines? Shouldn't we try to reduce latency?

Execution time and efficiency

- If we can reduce latency by a factor of 10, efficiency is drastically improved.
- A bit of surprize: Even the exascale machines still can show reasonable efficiency for 10k sec runs
- In other words, current exercise of very long runs are useful only to improve the performance from, say, 85% to 90%. Not much practical meaning.

How we "rank" machines?

Some people wants ranking. Several possibilities.

- Geometric mean of three numbers.
  - With similar network, larger machine would be still faster. Therefore, there would not be much effect on the current ranking.
  - Ultra-low-latency machines have big advantage. Shouldn't they?
- Take geometric (or harmonic) mean of the rankings in three categories.

Summary

- Current HPL considered harmful.
- HPCG considered more harmful.
- By measuring the performance of HPL for 10, 300, and 10,000 seconds, we can measure the network latency, bandwidth and peak arithmetic performance, of present and future big HPC machines.
- It is very important to put some measure for the latency, to guide HPC vendors (and those who responsible for the procurement process) to the direction of strong-scaling performance.
- We might still need one number for ranking. There are several ways to do so.