Answering the Really Important Questions Once and For All

(my humble contribution to this panel)

Thilo Kielmann
Thilo.Kielmann@vu.nl

COMMIT/

Disclaimer:

Exascale….

1-2 orders of magnitude bigger than
the limits of my imagination

When will we have a Linpack-exaflop machine?

- I don’t know. I am not a hardware guy…

- Why don’t you ask Satoshi?

When will we have a Linpack-exaflop machine? (2)

- We won't. (There is this resiliency thing…)

- But that is a bit pessimistic.

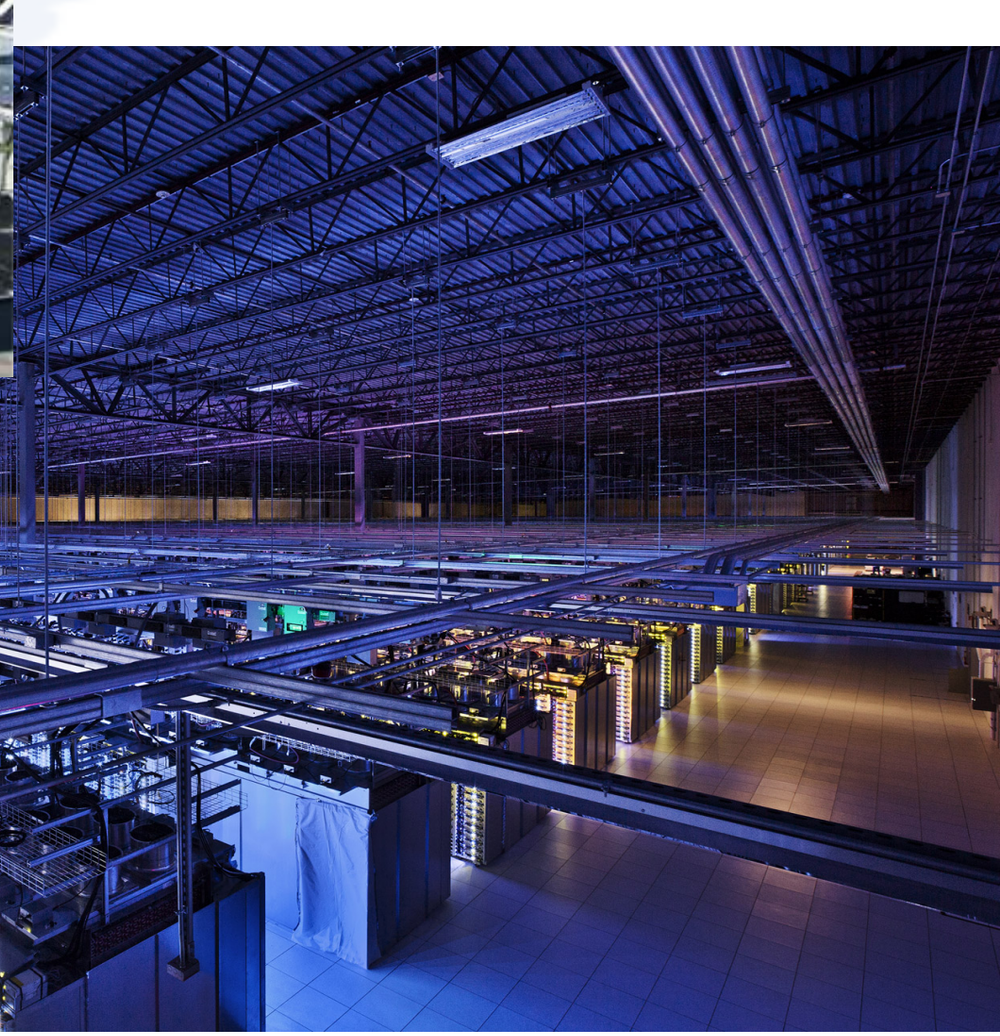
When will we have a Linpack-exaflop machine? (3)

- Well, we actually could have one, if we manage to handle this fault-tolerance issue…

What fault-detection, fault-avoidance, and/or fault-recovery mechanisms do you see in use now?

- working assumption:
  - MTBF(exascale) << application runtime
- Active replication: (3-fold)
  - I doubt we could afford wasting 2/3 of the machine, esp. at exascale
- Checkpoint/restart
  - All checkpoints are overhead (except for the last one)
  - Can we usefully checkpoint leaving large fractions of time available for the application? (more doubts)

But there are huge systems already in commercial production



What can we learn from the cloud?

- Data centers and supercomputers are siblings, separated at birth:
- Large numbers of machines
- Heterogeneous system
  - (certainly after the first hardware upgrade)
- Lots of data
- Large numbers of users who don't want to wait
- Energy consumption in the MWatt range
- Failure probability for the whole system ≈1 (?)

- And it seems to work well…

What is different?

- Build a reliable system from (many) unreliable components

- Do not expect the whole machine (data center) to sustain operation without failures

- Instead, have many, small units encapsulating failures, along with resilience mechanisms

  - replicate data (only)?

In 5-10 Years, HPC will be in the cloud!

- Maybe, most of it ??
- But only if the search engines, OS vendors, and book sellers of this world will get it right…
- PaaS and SaaS define the vocabulary for the future HPC-C programmer
  - how to define this? who determines that?
  - what kind of QoS/SLO can we define?
    - what kind of SLO do you want for a saxpy ?
    - Even for IaaS clouds, we live with “r3.8xlarge” rather than with Gflops x Gbytes-mem x Pbytes-persistent

Why isn't everyone programming in D?

- What is a programming language?
  - Wikipedia: *A programming language is a formal constructed language designed to communicate instructions to a machine, particularly a computer.*
- This is missing the point, the human factor:
  - *A programming language is designed for **humans** to communicate their ideas about how a computer should execute something.*

Why isn't everyone programming in D? (2)

- Humans are different by:
  - programming education (~~hobbyists~~ physicists, …, ~~professionals~~ computer scientists)
  - cultural background (Fortran, COBOL, Lisp, C, Python,…)
  - personal preferences (typed vs. untyped, imperative vs. declarative,…)
- Problems are different:
  - different nails need different hammers

Why isn't everyone programming in D? (3)

- This is a new formulation of a very old question, a nice red herring for computer scientists.

- See also: *NIERSTRASZ, Oscar. The next 700 concurrent object-oriented languages. Reflections on the future of object-based concurrency. Object composition, Centre Universitaire d'Informatique, University of Geneva, 1991.*

Why isn't everyone programming in D? (4)

- I don't believe in a single best programming language or model.
- We can agree to disagree, as long as everybody can solve his/her own problems in the his/her personal comfort zone.

What is the "Big Data" version of the Linpack benchmark?

- What about Graph500?

What are YOU currently doing to bring about the golden age of computational science?

Don't be shy.

- Ask me in the wine cellar!
  - I promise, I won't be shy…

COMMIT/