The evolution of the analysis ecosystem and its implications on platforms

Axel Naumann, 2017-03-23

CERN openlab Workshop on Compute Platforms and Software

Preface

- Problems for analysis are NOT
  - use last % of CPU
  - 100% efficient use of resources
- Problems for analysis ARE
  - usability and stability of platform
  - quick turnaround / agile development

Intro:

Status Quo, Context

Context

- Tradition:
  - experiment provides “blessed” data
    - physics groups filter into smaller sets (%)
    - physicists filter into laptop-sized sets (%*%)
- Last point is most difficult to steer, but is crucial for production of knowledge

Context (2)

- Physicists write analysis in C++
    - speed, correctness; experiments' software and default analysis framework (ROOT) in C++
- Physicists write analysis in Python
    - simplicity; excellent (compared to rest of the world) C++ binding
- Very resilient to language changes

Archetype of an Analysis

- Input data in (GB..TB), results (MB..GB) out
- Results peer reviewed, but generally not code; attempts to “force” archival process
- Iterative process of improving algorithms
  - deadline-driven, quick turn-around essential
  - reluctance to rely on complex systems, better use systems under physicist’s control

Embarrassingly //

- Data (and format) allows process parallelism
  - memory limit much less a concern for analyses
    - but results must be merged cross-process
- Multithreading requires knowledge of thread-safety
- Shared memory for merging, and separate memory for parallelism-safety?

Context Changes

Analysis Needs in 3y

- Challenges: Run 3 with 2x lumi
- Maybe tenfold data volume
- Diverse ecosystem: machine learning in R; Spark resources; python graphics
    - do we give up on our “default”, at the price of fractured analysis community with arbitrary ingredients, but benefiting from “industry standards”?

Analysis Needs in 5y

- Big challenges: HL-LHC (10x lumi)
  - 20-100 times data and simulated data
- Will “industry standard” go there?
- Can we store and move all that data?
  - or should we generate simulation on demand?

Platform Implications

Analysis Implications?

- Analysis traditionally parasitical, eats up as many resources as available
    - analysis will adapt to platforms, not vice versa
- But analysis is the currency of physics research
    - targeted resources will give optimal results

Memory

- Simplicity to program is paramount
  - homogenous address space
    - shared memory as bridge between processes
- More is better, but not crucial for analysis
- Persistent snapshotting of intermediary results

Storage

- Trade storage power for compute needs
  - we need to calculate what we cannot store
  - High demand might motivate paradigm shift for simulation
  - Multi-tier cache is fine: can prefetch
  - With parallel processing, single output becomes bottleneck!

Network

- Lowest-level cache; we can prefetch to hide latencies
- Expect increased role for remote graphics
  - low-latency RPC path
- No specific requirements

CPU

- Don’t expect the code to change: support generic code well
- Fewer high power cores per node remain desirable
- Decompression

Ideal Platform

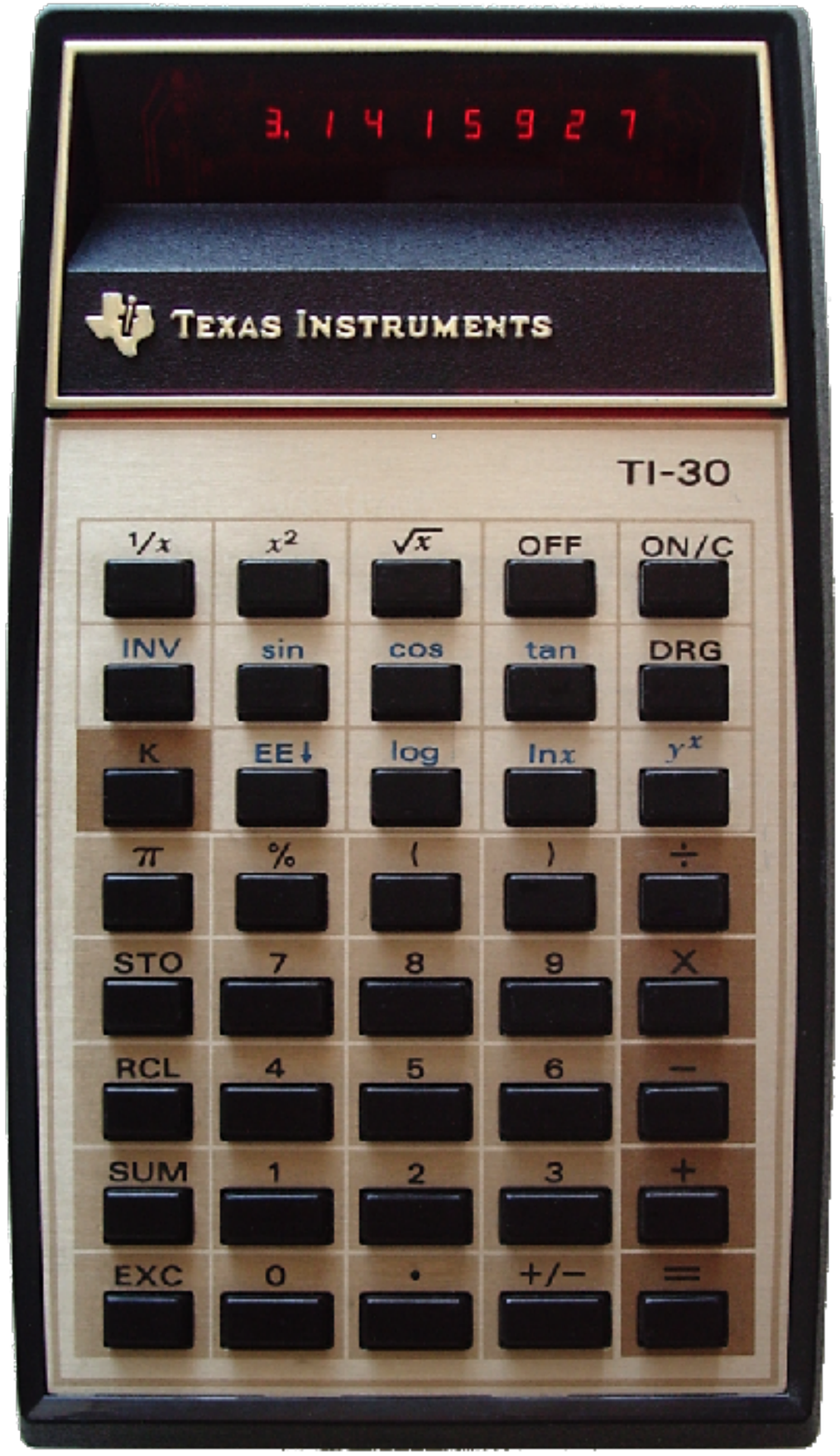
- Network bandwidth (laptops!) scales with compute power; data decompression is crucial

- RAM scales with cores

- Hardware accessible by “standard code” and “standard libraries”

  - big drive to open source, vendor independence

- Simple to program and robust, by design


3. 1 4 1 5 9 2 7
TEXAS INSTRUMENTS
TI-30
1/x
x²
√x
OFF
ON/C
INV
sin
cos
tan
DRG
K
EE↓
log
ln x
y^x
π
%
(
)
÷
STO
7
8
9
X
RCL
4
5
6
−
SUM
1
2
3
+
EXC
0
•
+/−
=

Software!

- Simplicity + compute bandwidth + C++ binding
- Open source
- Topics:
  - minimization, integration, machine learning
  - “transparent” parallelism: tasks? HPX? Ranges? ReactiveX? OpenMP? Co-routines?
  - composable, high-level code