Challenges for Scaling:

Co-Design for Memory Bottleneck, Power and Miniaturization

Group B

Members

1. Arata Amemiya	(RIKEN_R-CCS)
2. Bibrak Qamar Chandio	(Indiana U, PhD)
3. Marco Capuccini	(Uppsala U, PhD)
4. Kundan Kumar	(Indian Institute of Science, PhD)
5. Toshiya Shirakura	(Tohoku U, PhD)
6. Saurabh Gupta	(Indian Institute of Science, MA)
7. Hotaka Yagi	(Tokyo U of Science, BA)

Synthesis

- Large amount of data, that is mostly irregular and at times need to be processed at the edge, poses new challenges for scaling:
- Need for programming, architecture and power improvements.
    - Memory Bottlenecks
    - Portability (Miniaturization and Power efficiency)
    - Programmer productivity

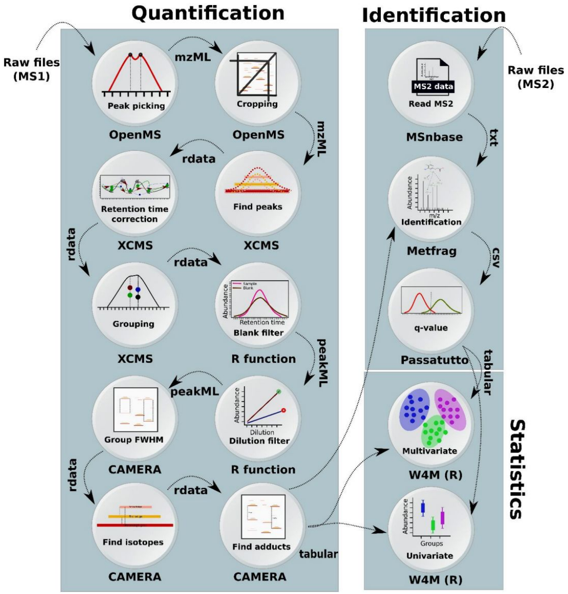
Motivations

- Democratizing Compute: (Bioinformatics & Smart Medical Systems)
  - Dataflow in Scientific Workflows
  - Intelligent Medical Systems Real Time Processing
- Scientific Simulations: (Quantum physics & Weather Forecasting)
  - Multi Precision Arithmetics
  - Data Assimilation & Learning
- Memory Acceleration: (Graph Processing & Machine Intelligence)
  - Non-von Neumann Architectures
    - Continuum Computer Architecture
    - Neuromorphic

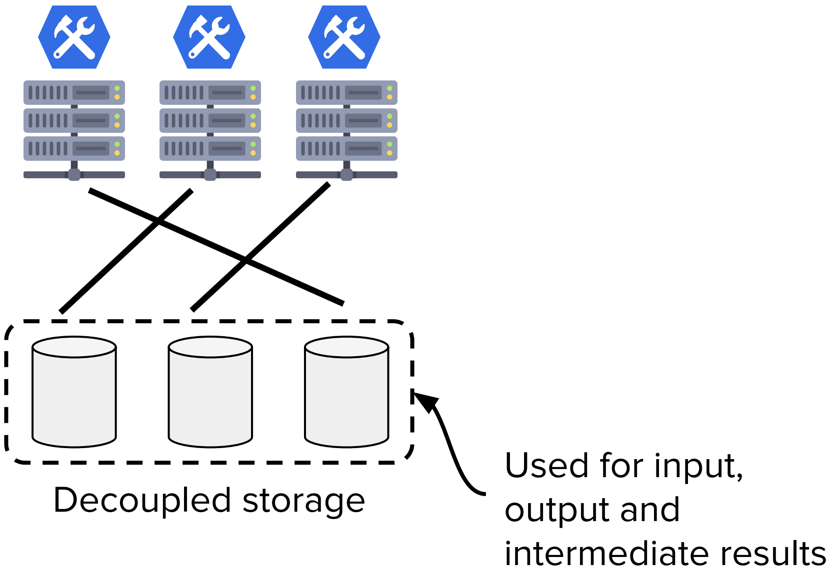
Problem Domain: Scientific Workflow with Containers

Omics (genomics, metabolomics, proteomics), machine learning pipelines, virtual drug screening

Scientific workflows

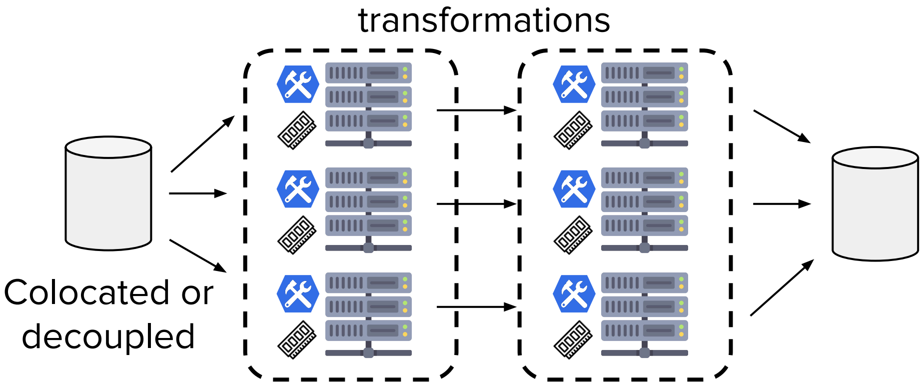


Problem: network contention



Solution: Dataflow programming model

MaRe 🐳 https://github.com/mcapuccini/MaRe



Memory is used for intermediate results.
How move data to/from containers?

- UNIX pipes
- Memory-mapped files
- Tmpfs

High-level API hides parallel computing challenges

- User productivity

Scales on cloud and commodity HW

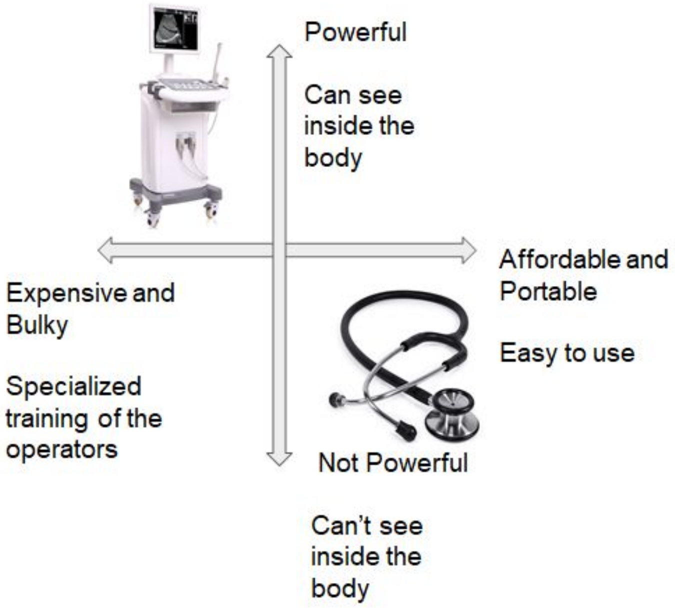
Problem Domain: Biomedical Diagnosis

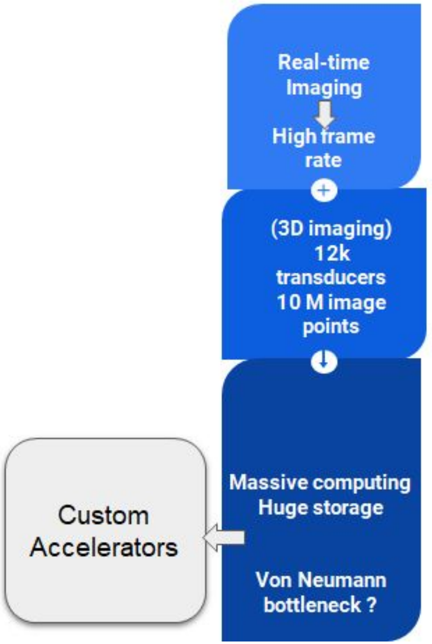
- Processing massive streams of data is an important problem in Biomedical diagnosis systems.
    - Biomedical diagnosis involves real time signal processing
    - A large number of transducers used, which generate massive data
    - Signal processing algorithms require huge memory to store pre computed coefficients
    - Accessing memory makes system performance slow : a bottleneck in real-time diagnosis

Example -

3D Ultrasound imaging requires 50 GB LUT (Lookup tables) space

Solution: Biomedical Diagnosis

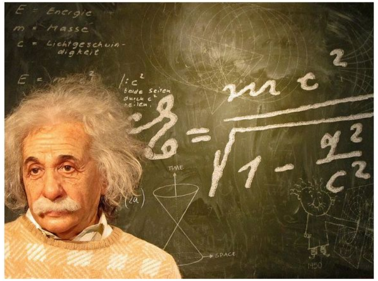




- Exploring sparsity of the data : compressive sensing
- Customized hardware : parallel computing
- On the fly computation : reduced memory access

Problem Domain: Quantum Physics

Numerical calculation for quantum physics



①What is the presence problem about quantum physics ?

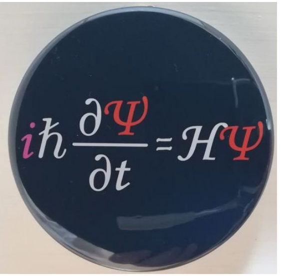
②Making program for numerical calculation

Considering computation time and capacity of files

Einstein equation

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu}$$

Schrodinger equation



Problem Domain: Weather Forecasting

Data size issues in data assimilation

Observational data size issues:

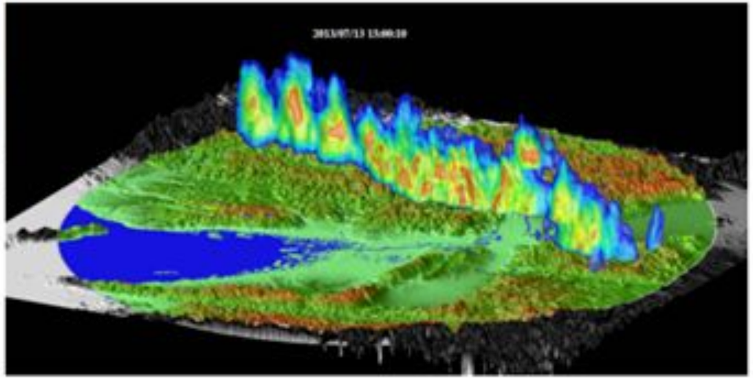
Real-time finescale weather forecast requires much observational data input

- conventional techniques (radar, satellites) with higher resolution
- new data sources (vehicles, portable devices)

Fast computation and data transfer are both essential

Possible solutions:

- improved pre-processing schemes



Problem Domain: Linear Algebra

Multi precision arithmetic

Double-Double and Quad-Double arithmetic uses the combinations of double precision numbers. # of operations would become large.

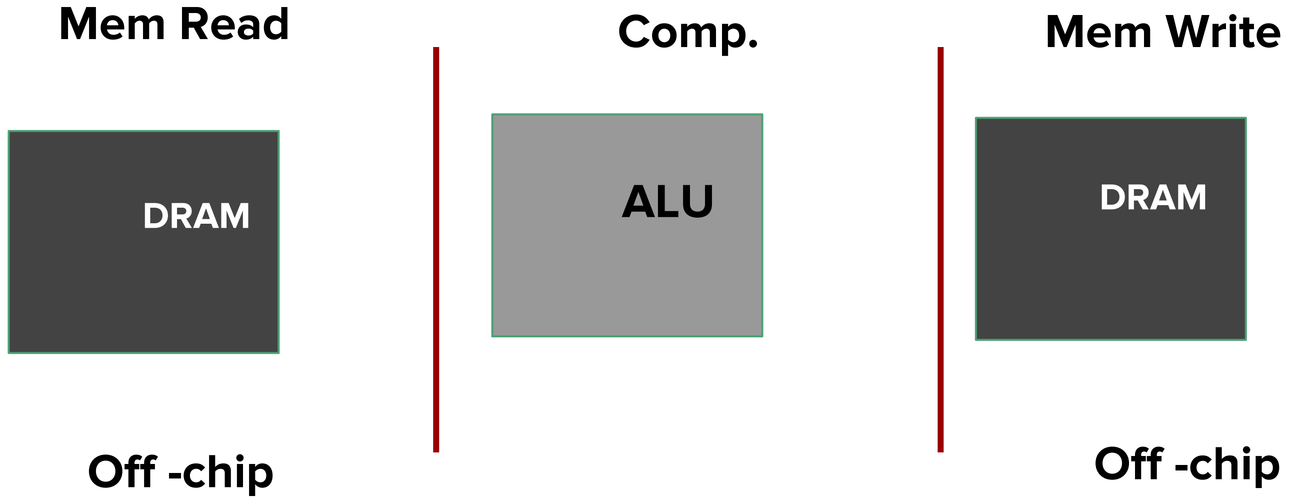
In the conventional laptop computer,

- Without parallelization, a kernel (BLAS 1 2 3) is computation bottleneck.
- With parallelization(FMA, SIMD, OpenMP), some kernels are memory bottleneck.

Parallelization have memory performance constraint for some multi precision kernels.

Problem Domain: Machine Learning

Memory Access - Bottleneck for DL applications.



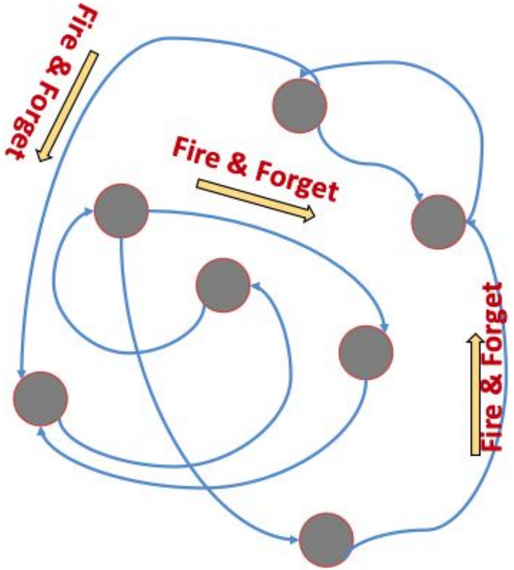
1. DRAM access: Data movement DRAM to ALU is expensive.
2. Mapping data-flow over the architecture: Memory hierarchy to computation units.
3. For DL application training and inferencing, loading huge data for training affects the training time, which may be critical for many real-time applications.

Solution: Machine Learning

1. Data compression to reduce the storage and movement.
2. Network pruning e.g based on magnitude of weights.
3. Reduce precision for computation: (Floating point -> Fixed point): 8 bit int used in (Google TPU).
    a. Binary weight, ternary weight..
    b. Non linear quantization (Log-domain)
4. Improve the reuse of data and local (computational) accumulation.
5. Exploit sparsity in the computation map: skip memory access and compute for zero.
6. Reduce operation while mapping DNN to matrix multiplication, example using FFT.
7. On-chip memory partition, putting memory and processor on same silicon substrate, increase the memory Bandwidth.
8. Moving from temporal architecture (SIMD) (MEM-> REG File -> ALU -> control) to Spatial architecture (more advanced for memory accessing) (MEM -> ALU).
9. Advance memory techniques: Stacked DRAMs and non-volatile memories.
10. Explore possibility of neuromorphic computing with asynchronous operation.

Problem Domain: Graph Processing

- Graph processing generally involves:
  - Low FLOP to Byte ratio
  - Irregular data access pattern
- Bulk Synchronous Model (BSP) leads to under exploitation of the large inherent parallelism that is naturally available in graph structures.
- Think like a Vertex, asynchronously:
- Send an active message asynchronous (fire-and-forget).
- No DAG. Because there could be cycles in the graph.
- We implement Dijkstra–Scholten algorithm for termination detection

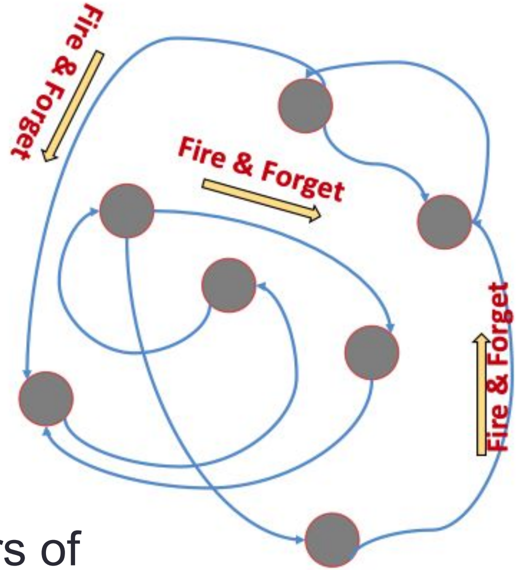


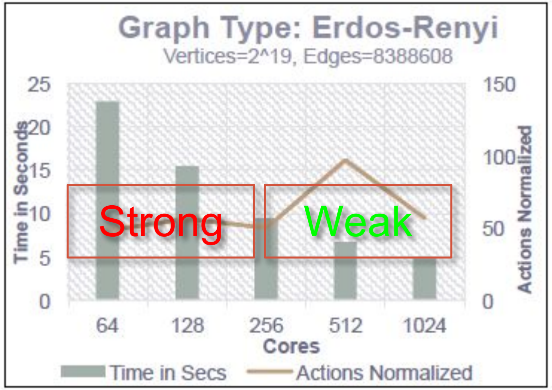
Problem Domain: Graph Processing

```
diffuse(vertex v, int distance):
  if v.distance >= distance:
    v.distance = distance
    for u in v.neighbors:
      diffuse(u, v.distance + u.weight)

SSSP(vertex src):
  src.distance = 0
  for u in src.neighbors:
      diffuse(u, u.weight)
```

Listing 1: Asynchronous SSSP

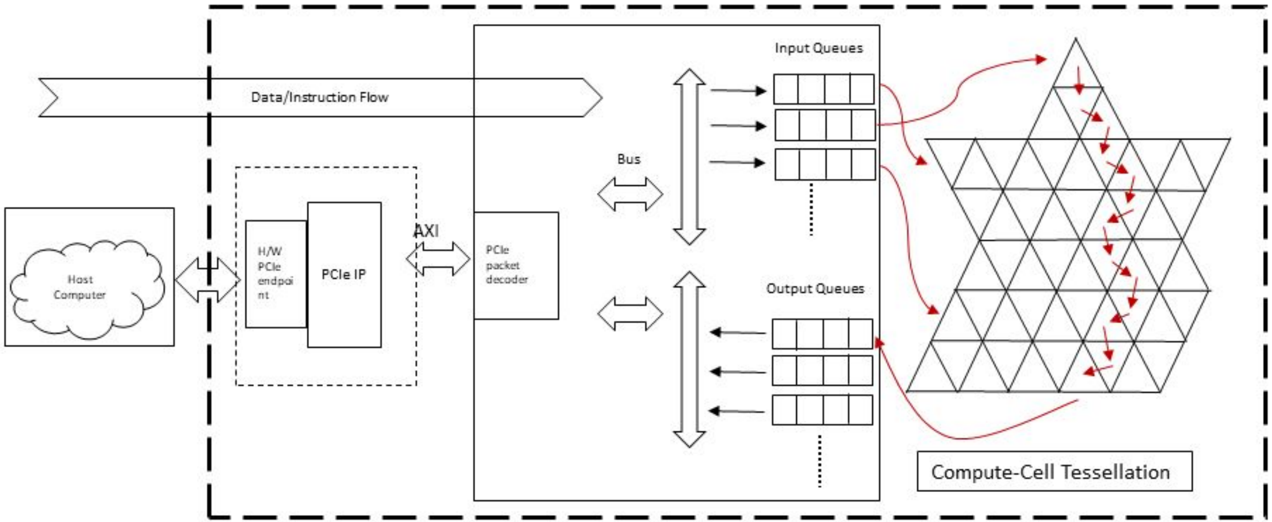




Presents both behaviors of Strong and Weak Scaling: Transcendental Scaling

Problem Domain: Graph Processing

- Continuum Compute Architecture is a new class of non von Neumann architectures.
- Offers fine grain parallelism.
- Small compute cells organized such that it creates an active memory.
- Low Power
- Less space footprint



Conclusion

- New Challenges posed by Big Data
  - Irregular memory access
  - Memory bottleneck
  - Latency sensitive
  - Low Power requirements
- Solutions:
  - 3D stacked Memory
  - Non-von Neumann architectures: send work/compute to memory and process there
  - Custom hardware for inference (and other compute) ➔ less power and less areas footprint, critical for portability
  - Dataflow-oriented workflows
    - Programmer productivity
    - Auto optimizations (lazy evaluation, concurrency, locality optimization)