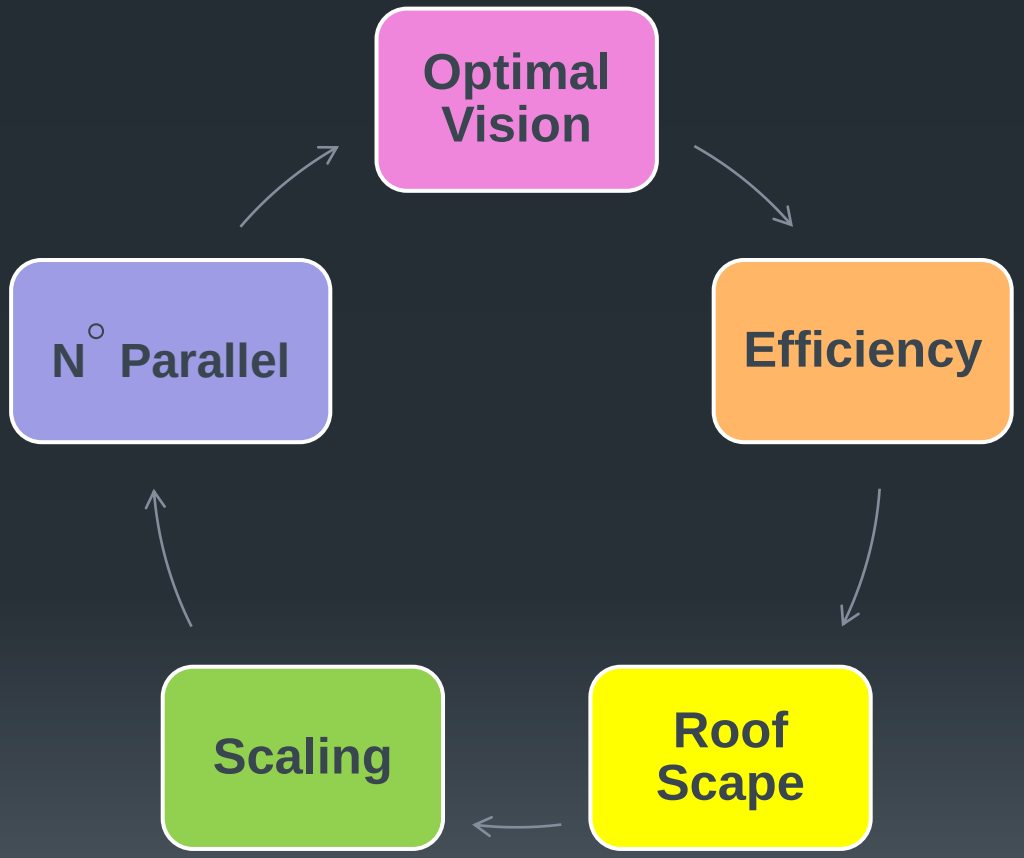
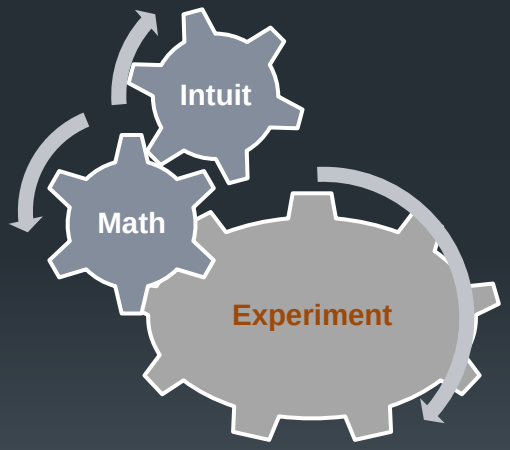
A Vision (vs view) of Optimal (vs potential) SDP Compute Node Configuration

- a lot more advanced than a year ago
- not always traditional wisdom compliant

C4SKA at AUT
2019-02-14 & 15
tn@compucon.co.nz

Contents
a co-design exercise



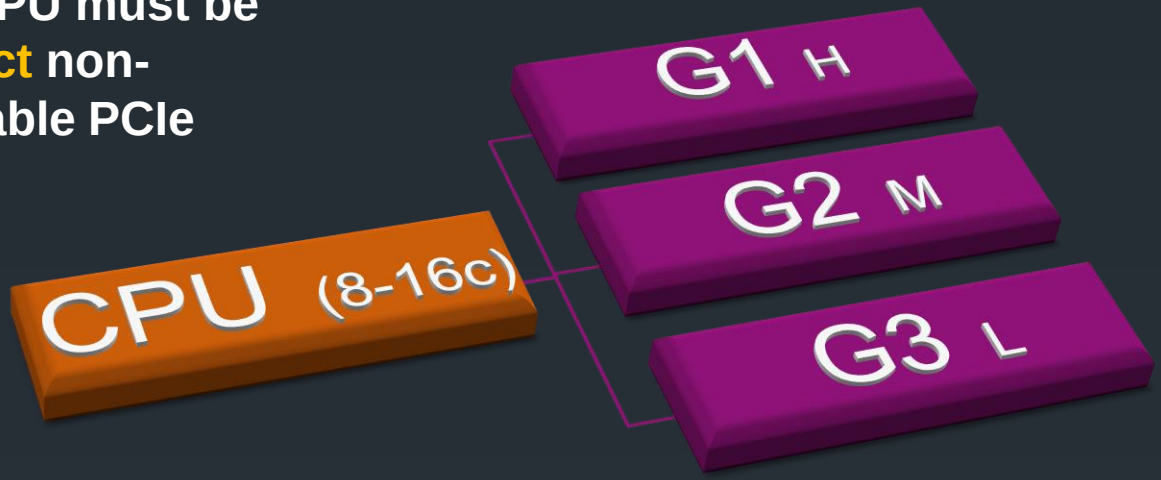


Benchmarks emphasised by 3 Turing laureates

1. Vision of a compute node

optimal >> 10% overall SDP efficiency

Each GPU must be on direct non-switchable PCIe links



3 cores for 3 GPU scheduling, rest for science computing in collaboration with a GPU

Over half of SDP science algorithms are memory intensive. Even for compute intensive apps, G1H of 3x price of G3L may deliver 2x performance only

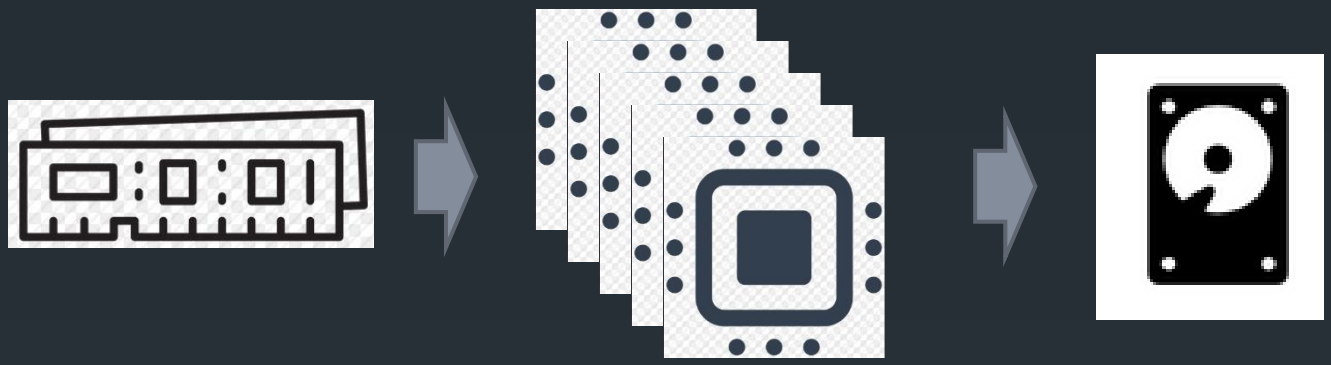
2. Compute Efficiency

Interim target for optimization

Visibility from CSP

Imaging Pipeline

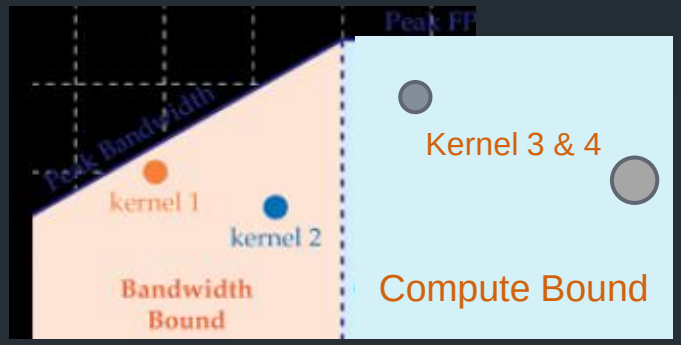
Local Sky Model

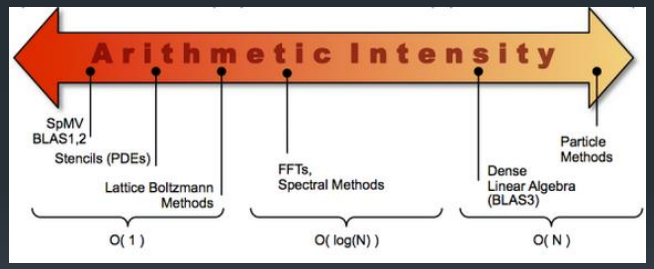


- Each application has its own compute efficiency
- How to estimate an average efficiency of pipeline for hardware resource estimation?

3. Roof Scape

Foundation of co-design framework





Credit of Roofline Model and OI definition to UC Berkeley and images to KAIST and Berkeley Lab

a. **Compute Efficiency = Roofline Efficiency x Programming Efficiency**

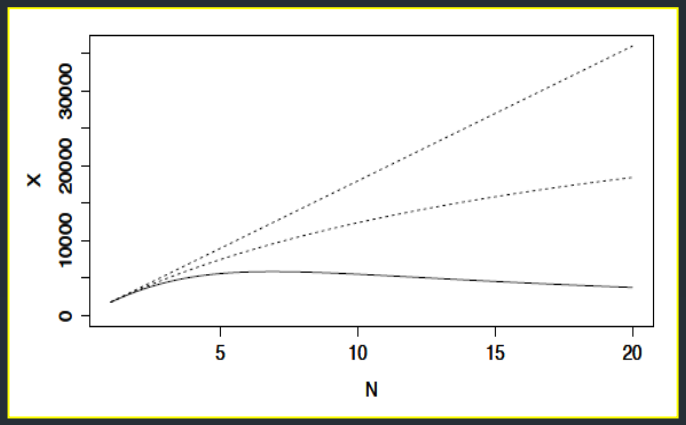
b. **Composite Operational Intensity of Pipeline**

$(OI)p = \Sigma [Wi * (OI)i]$ if $(OI)i < (OI)r + \Sigma [Wi * (OI)r]$ if $(OI)i >= (OI)r$

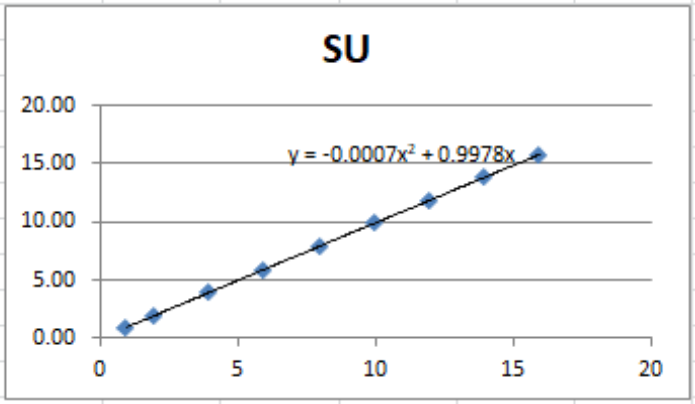
- **SDP Size = 259 PFLOPS DP / (X PFLOPS DP * CE_p * Y) where Y is scaling efficiency**

4. Scaling Efficiency

Loss of compute efficiency



E7	60k
ncpu	SU
1	1.00
2	2.01
4	4.02
6	5.95
8	7.94
10	9.89
12	11.87
14	13.83
16	15.82



LHS

- **Speed Up = N / (1+a*(N-1)+b*N*(N-1))**
  - **Where N is number of compute nodes or workers**
  - **Credit to Gene Amdahl and Neil Gunther for 2 applicable laws**

RHS

- **Cholesky Factorization of 60kx60k in 100x100 blocks by EYPC 7351P with 16 cores**
  - **Credit to StarPU for scheduling (& optimization)**
  - **Almost linear scaling for up to 16 cores (98% SE) despite high data dependency**

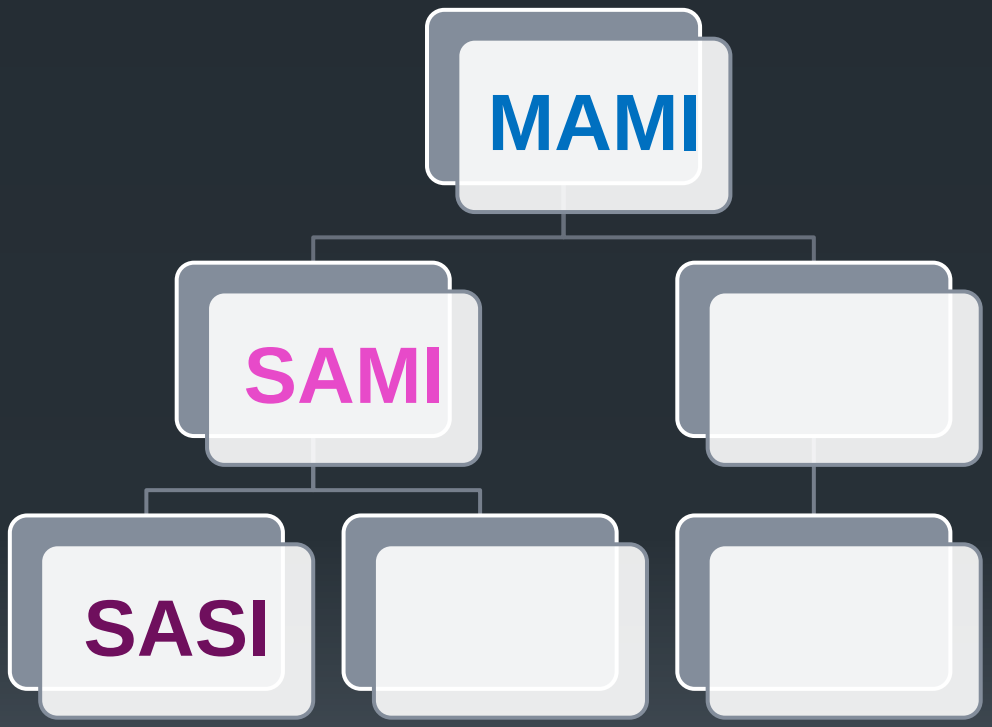
5. Degrees of Parallelism

A hierarchy

More degrees at pipeline level (independent splits) for Optimising Scaling Efficiency

Implementable with (view only)

- Software in 2020
- Hardware 2025



S for single, M for Multiple, I for instance

6. (Bonus Slide) Illustration

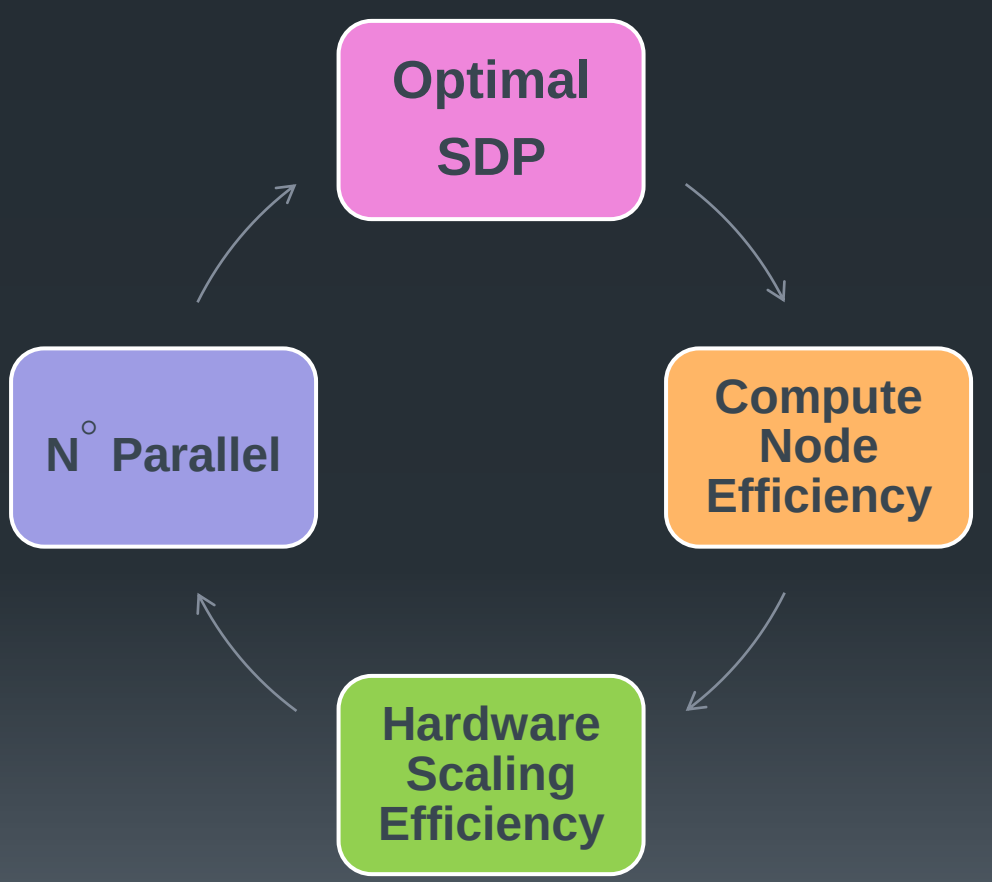
Table 1/2. Cholesky Factorization, Scheduler = dmda, CPU = 16 core E7351P, GPU = 1050Ti

Problem	Algo	E7351P Only			1050Ti	Hybrid 1 (E7351P,GTX 1050Ti)		
Size	OI	1 core	16 core	SU 1/16	Only	1 core	14 cores	SU 1/14
10k, 20	42	15,861	1,260	12.6	585	438	358	1.2
10k, 100	9	24,679	2,608	9.5	3035	3024	2364	1.3
30k, 20	125	411,677	31,004	13.3	6192	5713	5589	1.0
30k, 100	25	415,934	27,629	15.1	9133	8240	6964	1.2

- Cholesky Factorization written in C
  - Block approach speeds up about 5 times (white columns)
  - in 4 problem sizes & 3 hardware arrangements
  - Figures are runtime in millisecond
- Findings
  - More Cores are needed if only CPU is available (yellow)
  - GPU is not always faster than CPU (brown)
  - CPU + 1 GPU was optimal in hardware utilisation (light green)

Vision 2021 at C4SKA-2020

revealing efficiency on a bigger scale



SDP ARL-based science pipeline C code