How Pascal And Power 8 Will Accelerate Counterparty Risk Calculations

GTC Europe 2016

28-29 September

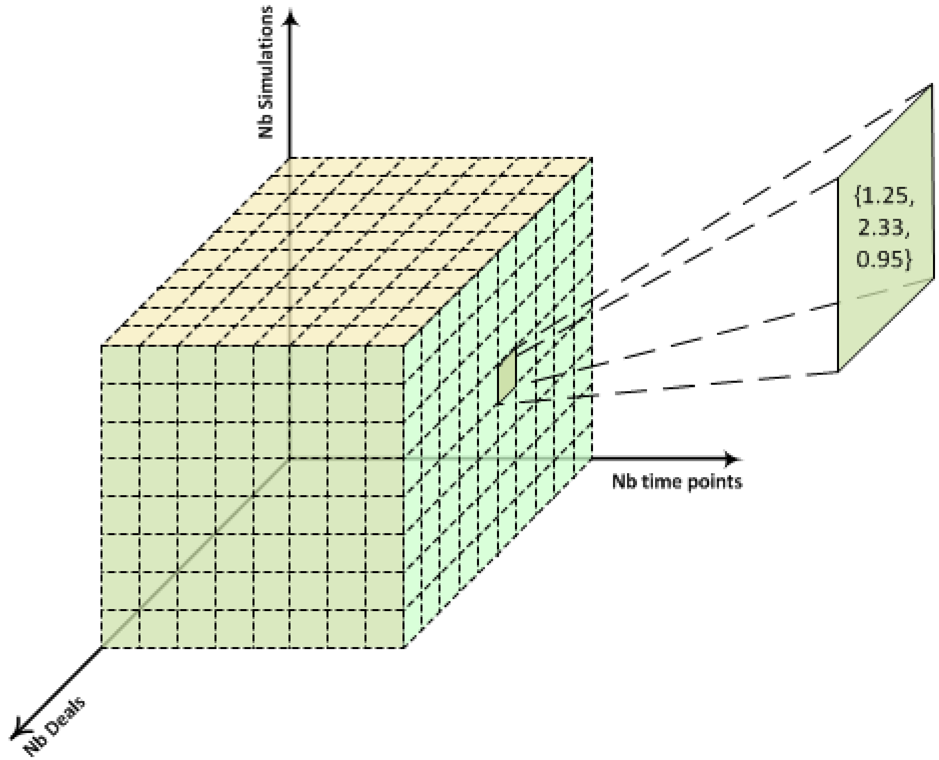
Amsterdam

Summary

- Counterparty Risk
  - Massively Parallel problem
  - From Big Data to Massive Compute
- Quantitative Libraries
  - Performance vs Code Flexibility?
- The DAG
  - Pricing algorithm as a Directed Acyclic Graph
- DAG Shapes and Sizes
  - New degrees of freedom with DAG chunks
  - NVLink and the DAG
- Expectations on Pascal

Counterparty Risk

- Exposure Cube
  - Calculating all points
  - Aggregation along deals (simulations x time points)
  - Sorting and aggregation along time points for risk measures Exposure



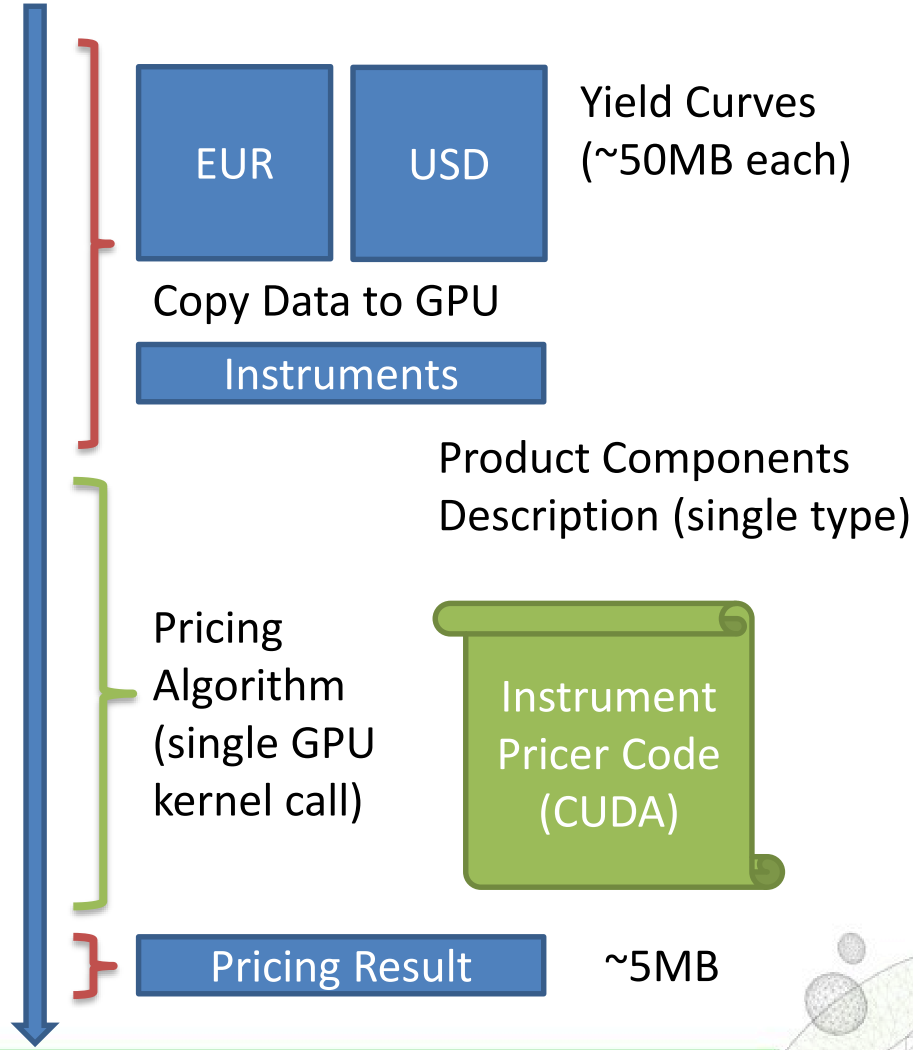
- Problem Dimension
  - 1,000,000 x 5,000 x 400 time points 2 trillion calculations (16TB of doubles)
  - This is one run…. We need a few hundreds

Counterparty Risk Calculations

- Risk Calculation
  - Large Problem: hundreds of servers, hundreds of TB, dozen of Databases…
  - Problems: Cost, Scalability, Maintenance…
- GTC 2013
  - Presentation of results and performance at GTC 2013
  - Live mid-2014
- Quantitative Library
  - Source code written in C# (popular amongst quant analysts)
  - Hybridized in CUDA/C and C++/OMP
  - Compiled to native target (GPU/CPU)
  - Used in a distributed Java application: Symphony, Coherence, Cassandra, Splunk

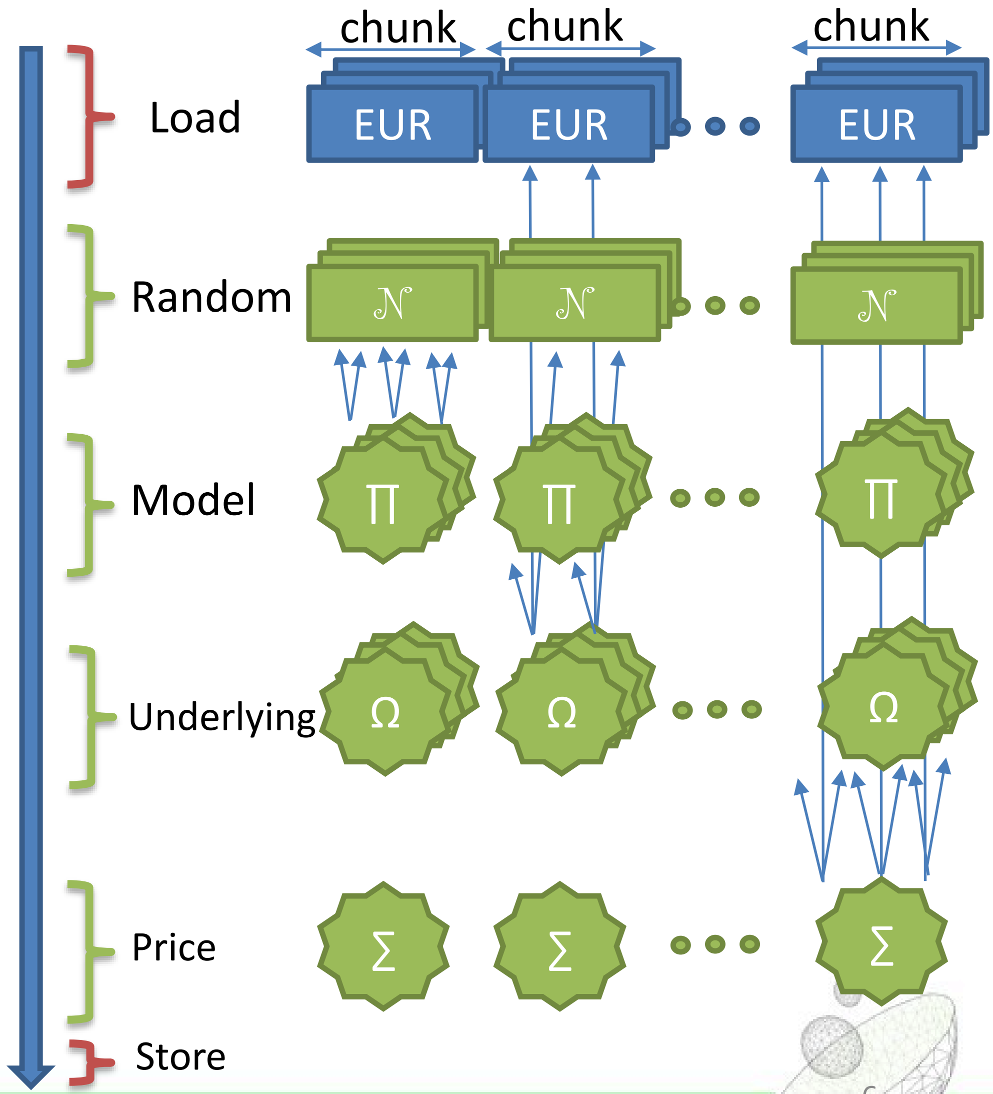
Quantitative Libraries

- Quantitative Library for IRFX (live 2014)
  - Complexity is low/Medium
  - Code generated is CUDA/C and C structures
  - It is all about pricing (simulations are generated once beforehand)
  - Pricing fits in single GPU kernel method

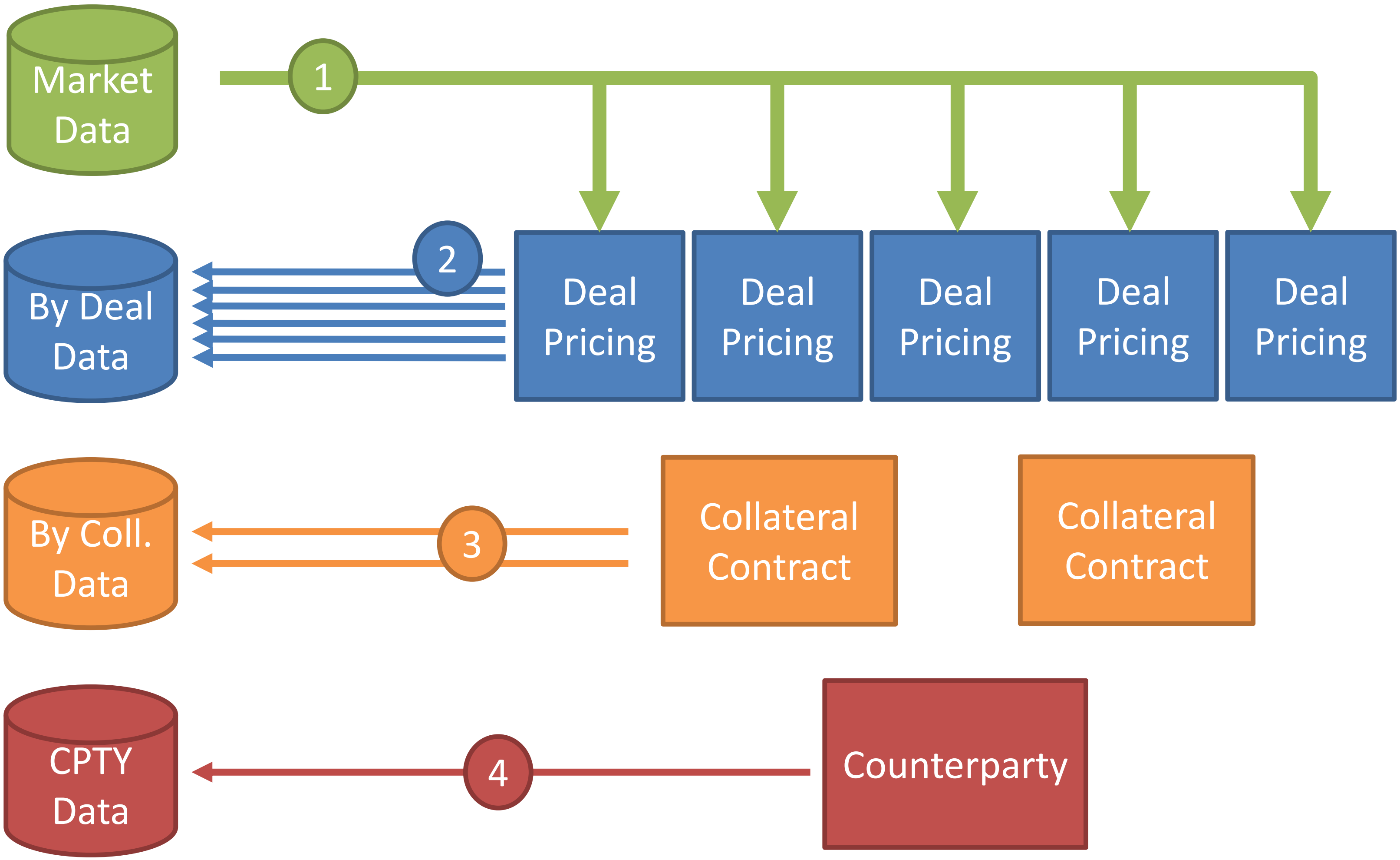


Quantitative Libraries

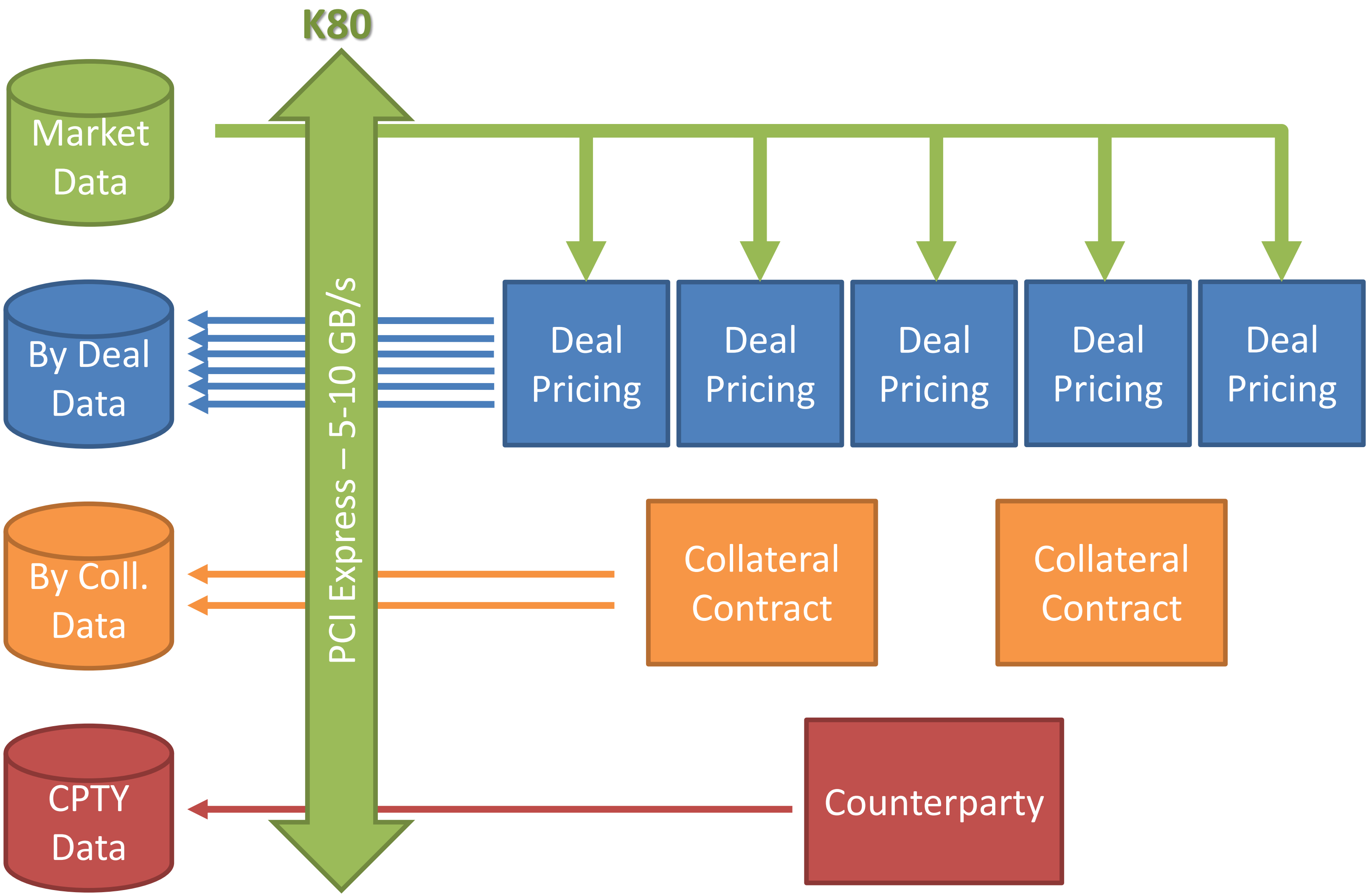
- Quantitative Library for Equity/Commodity (2015)
  - Complexity is medium
  - Need an object oriented model
  - Number of MC paths: x2.5 IRFX
- Quantitative Library for Credit and Repo (2016)
  - Complexity is high
  - Simulation and Pricing need to be interlaced
  - Number of MC paths: x2 EQCM, x5 IRFX



Computation with Directed Acyclic Graph



Computation with Directed Acyclic Graph



Computation with Directed Acyclic Graph

DAG is built for a chunk of simulations N = k*32

CPU Memory Footprint		GPU Memory Footprint – working set
Up to 20 GB / stress scenario	Load Market Data From CPU RAM	45 kB / DF / sim
[Optional] 15-30 MB / deal	Price Deal A · Price Deal B · Price Deal C	3 kB / deal / sim
15-30 MB / contract	Collateral Contract · Collateral Contract	3 kB / contract /sim
15-30 MB	Counterparty	3 kB / sim

The larger the chunk of sims, the more parallelism

Computation with Directed Acyclic Graph

- Performance bottleneck
  - CPU to GPU Bandwidth
  - Variable between (GPU)
    - Memory-bandwidth bound
    - Memory-latency bound
    - Compute bound
  - GPU Memory bandwidth
  - GPU Memory bandwidth

Load Market Data From CPU RAM

Price Deals

Collateral Contract

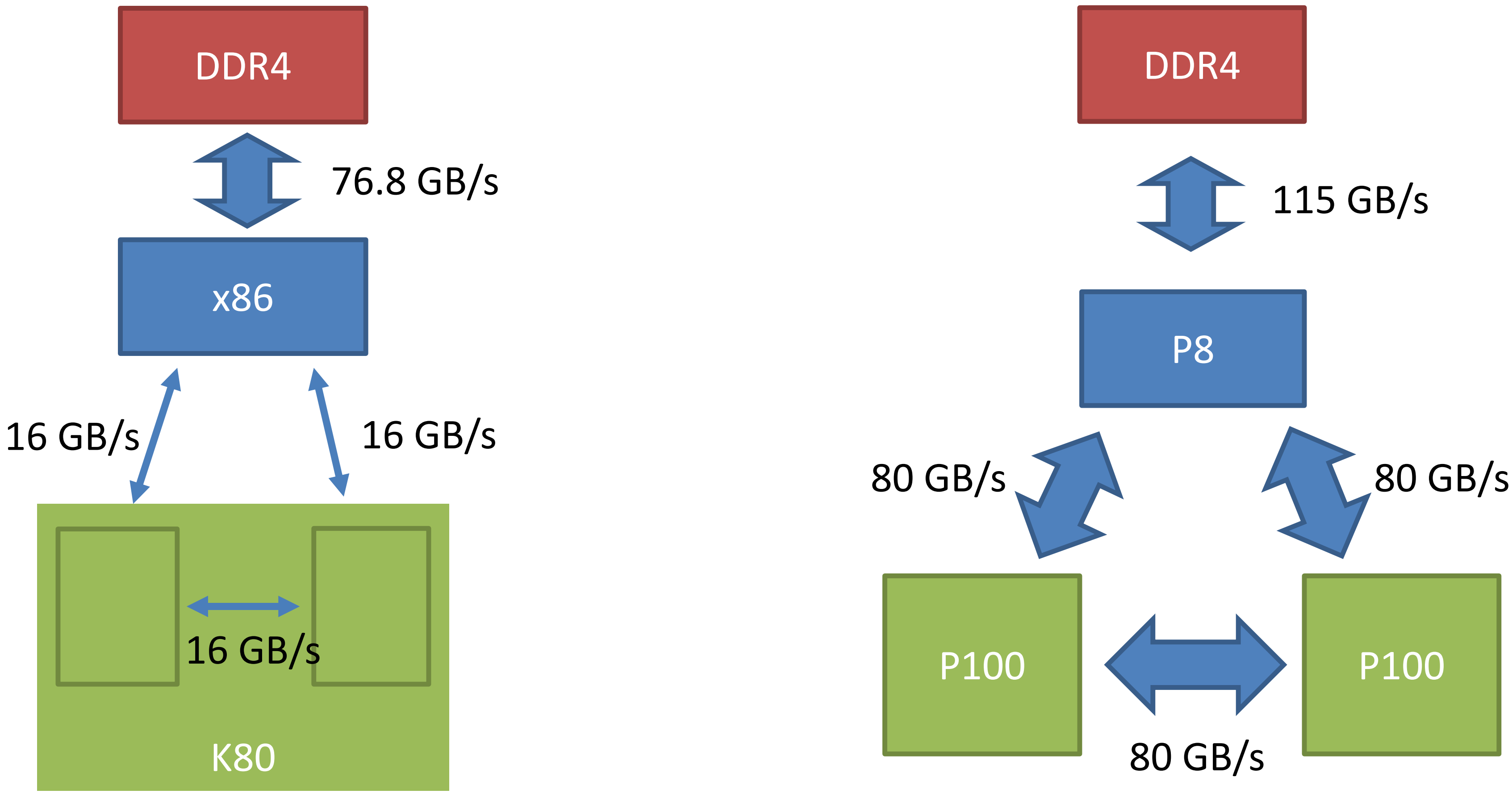
Counterparty

Computation with Directed Acyclic Graph

- Build a DAG of calculation nodes
- Working-set depends on GPU memory budget
  - Larger simulation chunks mean more parallelism
  - More deals mean more market data reuse

- NV-Link makes usage of system memory as intermediate buffer viable – new options
  - Market data resides on CPU and is not cached on GPU memory
  - Output buffers never allocated on GPU
  - Some intermediate buffers never allocated on GPU

NVLink Offers New Work Load-Balancing



More flexibility for work distribution: large chunks and DAG split amongst several devices

Using Page Migration

CPU Memory Footprint	DAG is built for a chunk of simulations N = k*32	GPU Memory Footprint – working set
Up to 20 GB	Load Market Data From CPU RAM	~~45 kB / DF / sim~~ pages migrated on demand
[Optional] 15-30 MB / deal	Price Deal A; Price Deal B; Price Deal C	**3 kB / deal / sim Performance cursor**
15-30 MB / contract	Collateral Contract; Collateral Contract	3 kB / contract /sim
15-30 MB	Counterparty	3 kB / sim

DAG Size Balance

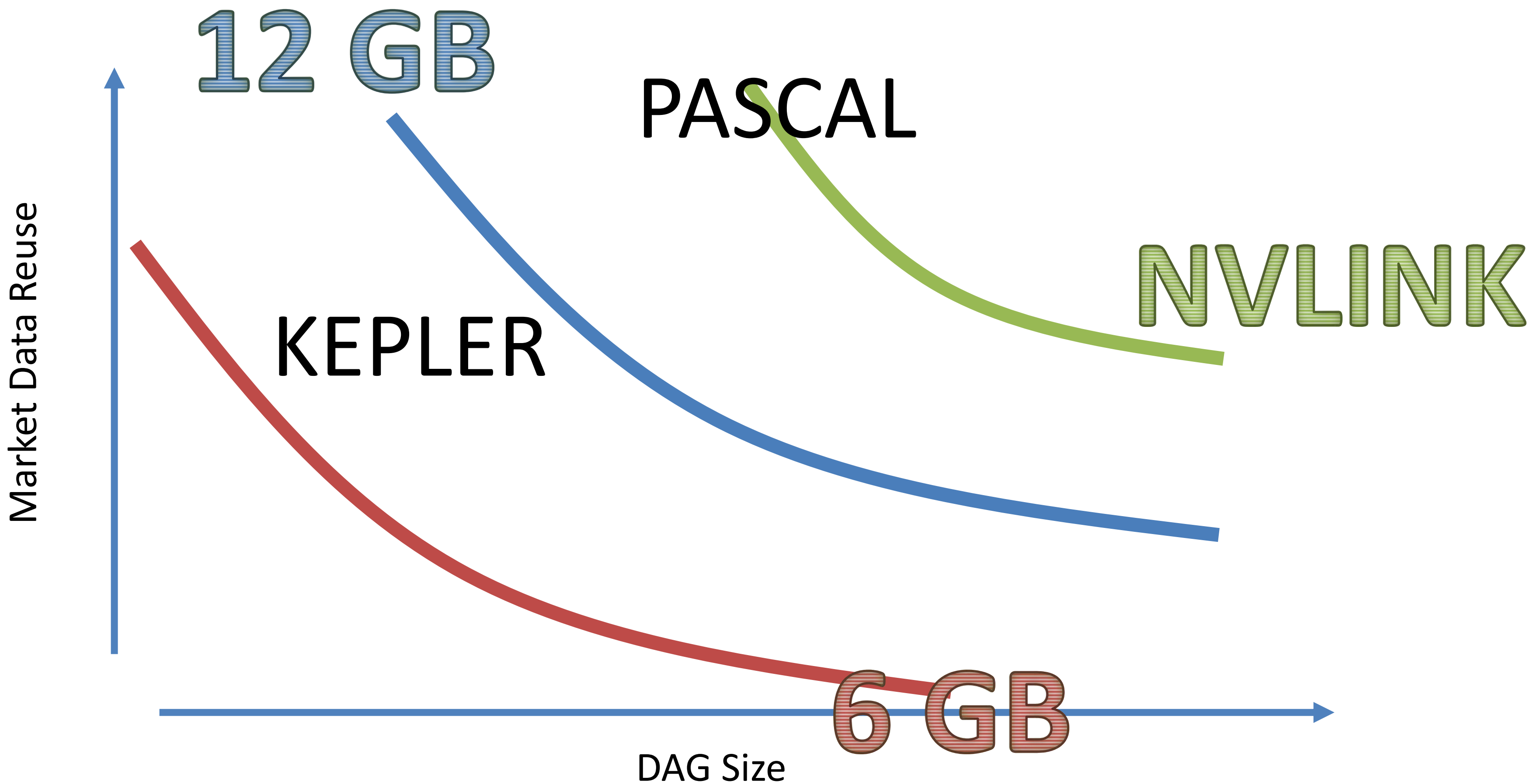
LARGE DAG (many deals)

- Benefits
  - Significant reuse of market data
  - Block-level parallelism
- Drawbacks
  - Smaller chunks mean lower parallelism
  - Yields performance penalty on large SMX from Kepler

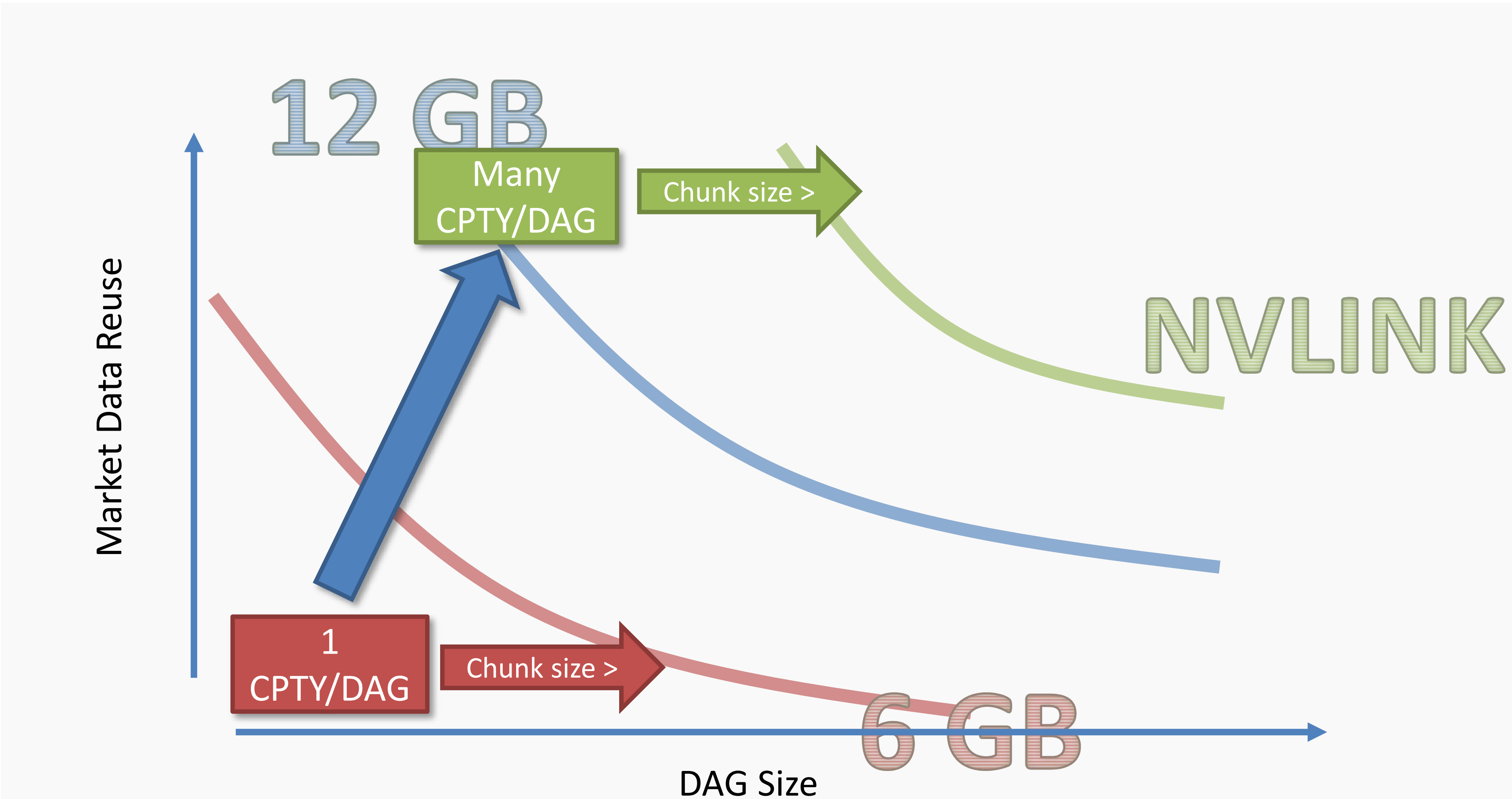
SMALL DAG (large simulation chunks)

- Benefits
  - Better parallelism
  - Lower memory-latency boundness (many blocks may work on same code)
- Drawbacks
  - Little reuse of market data
  - High performance penalty on Kepler as market data transfer is slowest

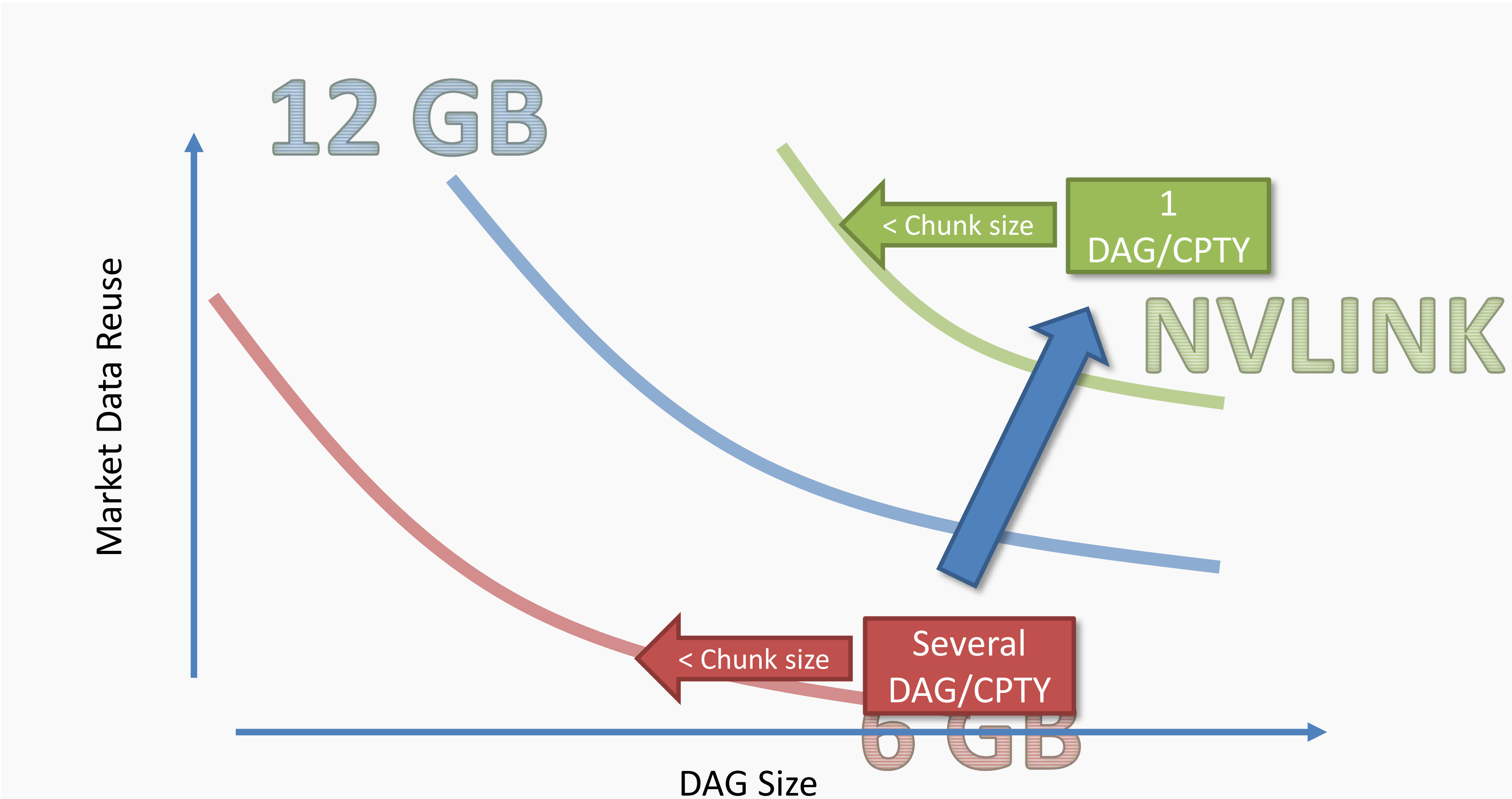
DAG Size Balance



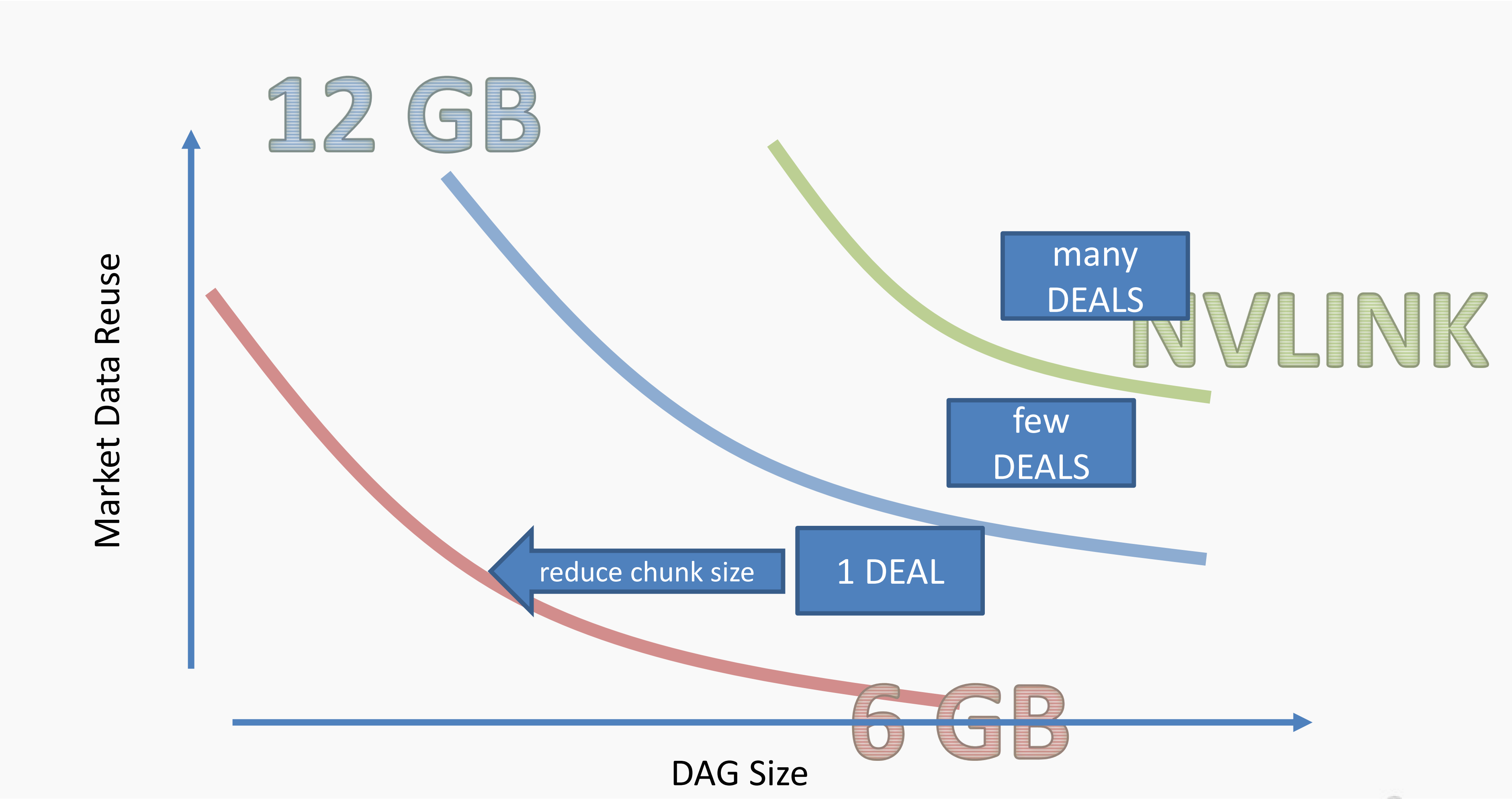
DAG Size Balance – Small Counterparties



DAG Size Balance – Large Counterparties



DAG Size Balance – Very Complex Products



Benefits of Pascal

Harwdare	K80 (1/2)	Pascal (Minksy)
FLOPS	1.45 TF	5.3 TF
GPU<->GPU	240 GB/s	720 GB/s
Memory Size	12 GB	16 GB
Interconnect	8x Gen 3	NVLink
GPU<->CPU	16 GB/s	80 GB/s
Local/extern	15	9
Watts	**150**	**300**
FLOPS/Watts	**9.66**	**17.6**
GB/s/Watts	**1.6**	**2.4**
Ext./Watts	**0.107**	**0.267**

- Small counterparties
  - Group counterparties (more data reuse)
  - Easier to reach local/extern peak
- Large counterparties
  - Coarser split (more data reuse)
  - Larger chunks (more parallelism)
- Very Complex Products
  - Larger chunks (more parallelism)
  - Coarser split (more data reuse)

Example Configurations

Metric	2 x Intel + 4 x K80	RATIO (vs Pascal config)	2 x Power + 4 x Pascal	RATIO (vs Pascal config)	2 x Intel + 2 x K80
Watt (TDP)	1600		1900		1150
GPU Compute GLOPS/W	7.25	**1.45**	10.53	**2.09**	5.04
GPU Memory GB/s/W	1.20	**1.26**	1.52	**1.83**	0.83
CPU-GPU Link (GB/s/W)	0.080	**2.11**	0.168	**3.0**	0.056

Early tests on an engineering Minsky sample with pre-release driver version illustrate an aggregate NV-Link read bandwidth of 120 GB/s, that is 75% of theoretical peak. In comparison, the best bandwidth obtained on two K80 is 22 GB/s which is 69% of theoretical peak.

System TDP: bi-socket Intel: 400W, bi-socket Power: 700W, K80 board: 300W, Pascal mezzanine: 300W

System and GPU Architecture Evolutions

- Quantitative Library Evolution
  - Started as a C-style Library…
  - …now an Oriented Object Library
  - On the fly simulation, with chunks, offer perfect scalability
- Nvidia Evolution
  - Cuda: CPU Memory in GPU Address space
  - Hardware: More Cores, More memory, More performance/Watt, Easier access to performance
  - *Evolution Handbrakes: Support different architectures in our data center (Fermi, Kepler and soon Pascal)*
- Pascal
  - High NVLink bandwidth changes the deal of host memory accessed by device

Thanks for your attention

http://www.altimesh.com