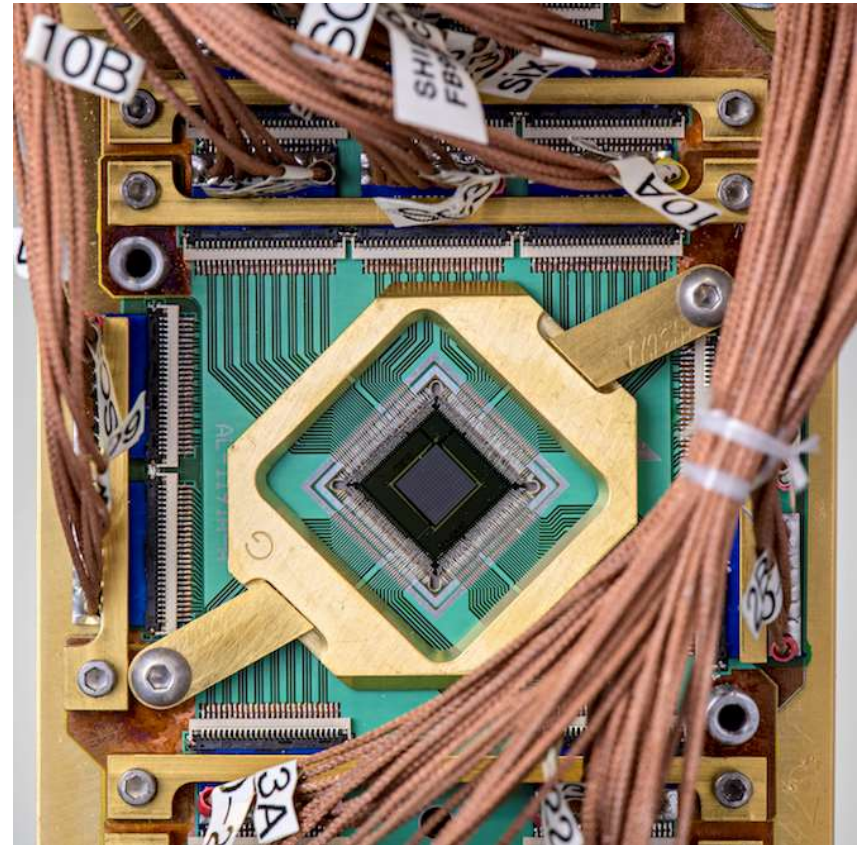




Scaling Asset Sustainment to Industrial Size

Aussie Schnore, Annarita Giani
GE Research

Qubits North America
09/24/2020



Industrial Scale

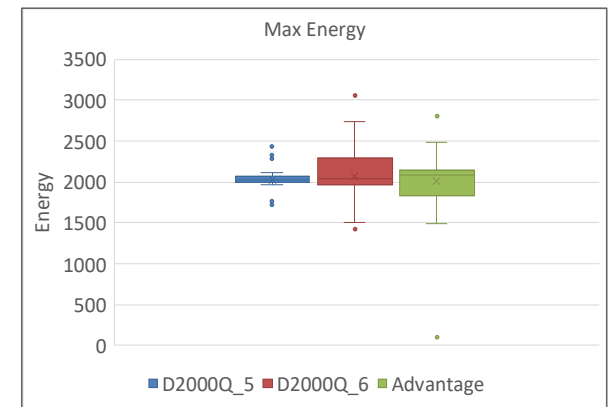
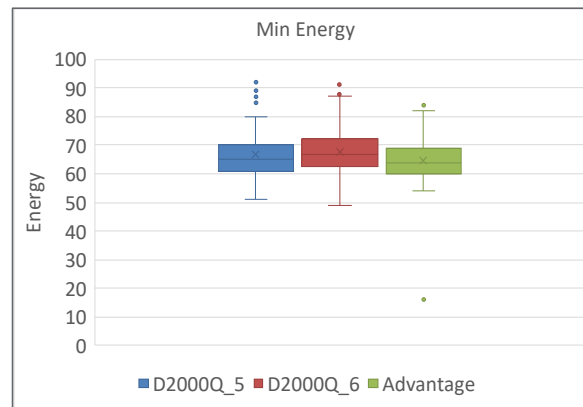
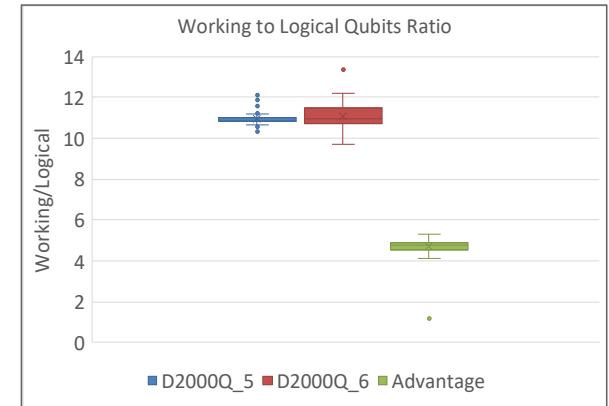
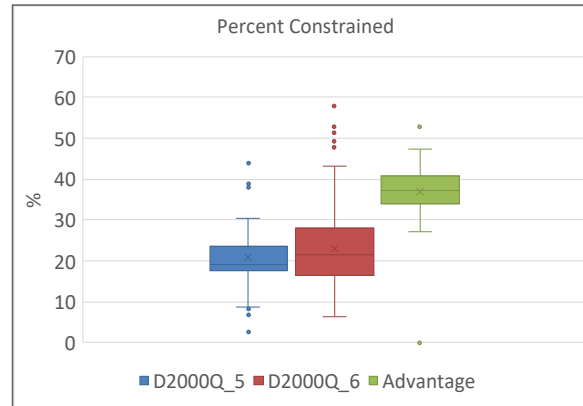
Domain	Number of Entities	Alternatives	Timeliness	Relative Value
Resource Allocation	15,000	$\sim 10^{6,500}$	1 day - 1 hours	\$100M+/year
Scheduling	39,129	$\sim 10^{110,000}$	30 days - 6 hours	\$100M+/year
Configuration	260	$\sim 10^{250}$	months	\$100M+/year

Classical computing is doing very well so you need to go after the tough problems.
Not a lot of low hanging fruit.

Advantage – Improved Access to Qubits



We had to tell our visitors that you couldn't use all the qubits to solve the problem



Perfect Embedding

At Qubits 2019

Working Backward to Full Scale Problem

Started with small perfect embedding

Constraint Edges:
(A,E),(H,D),(B,F),(G,C)

Resource Edges:
(A,H),(A,G),(A,F)
(E,D),(E,C),(E,B)
(H,B),(H,C)
(D,F),(D,G)
(B,G),(F,C)

$2^4 = 16$ answers

Creating a problem with perfect embedding

Big Problem

927 repairs
Two paths each

$2^{187} =$

```

11340208080416078010561134548205467019074836471008201100807
8821288015899120281056080313477703003347680852212754007331
88209884742333316272304207801054227901979480809002121460711
24873720026441181271202602108100100203070017803646001201
949800546543017511614811020728
    
```

Answers
 $\sim 10^{57}$

- Red lines are constraint couplers
- Blue lines are resource sharing
- Used 1854 qubits (~90%) and 4917 couplers

Creating a problem with perfect embedding

Sampled distribution from $\sim 10^{297}$ answers

- 4567/5000 constrained samples
- Need to find way to do importance sampling to focus on tails at this scale
- Over 3 runs no redundant samples (need to try more)

Results

Next Pegasus

- With 5640 qubits
- 40484 couplers

Based on a calculated embedding with 94% qubit utilization

2674 repairs with two paths each sharing 37810 resources

$2^{194} =$

```

8991078245110224958806709131320509040188844718148576442556
076397874858968699128115888518942424876388439125248127949436
749398709001260899265147310283735615949904081871251233014
18355558850632497468965828367576813662779553388041939
8111864475744035871897291823541731575179592626603134460225
879106845450712328987462641159442388157681100781877134870656
7369020054494712280337919050267736669978620085666913742131
20161630433091203080132511602164639049490332414335324841
1898468384281132113897882842851189508917809916637358017124
71077978130044932082531462178918860617415508780731559604
5660009784113842327343572945869518474733174142646279027
63942434677033876981863182486957768189482956870774556937
329432282187720762328496973878
    
```

Answers
 $\sim 10^{59}$

Estimate based on 90% utilization
1692 repairs with three paths each

$3^{192} =$

```

134600028650204038848888004100207251342247340748210749588
30979612947252797120123725186152852190759077088615643655
2544218822478800312080338809721318403008076624240818511965
2817124821350365616209938892631161313815149925404433485864
150211031889161415151361381794781460781678815502070421210837
7895671724540097463443878762024043520013114649724268570887
384468680087607821923403883110332047630831808204081082857
4875468686881384991421102323286664288160388819984491927
529978747429344444762224909427433978999121444048524272736
55305521510260660205607705602954251594095144003388753151
0467537311522967627456646225818424210571440373536211983383
0717817888481811038851057946404258138944879323101247089791
64456462949373152631588584117083541
    
```

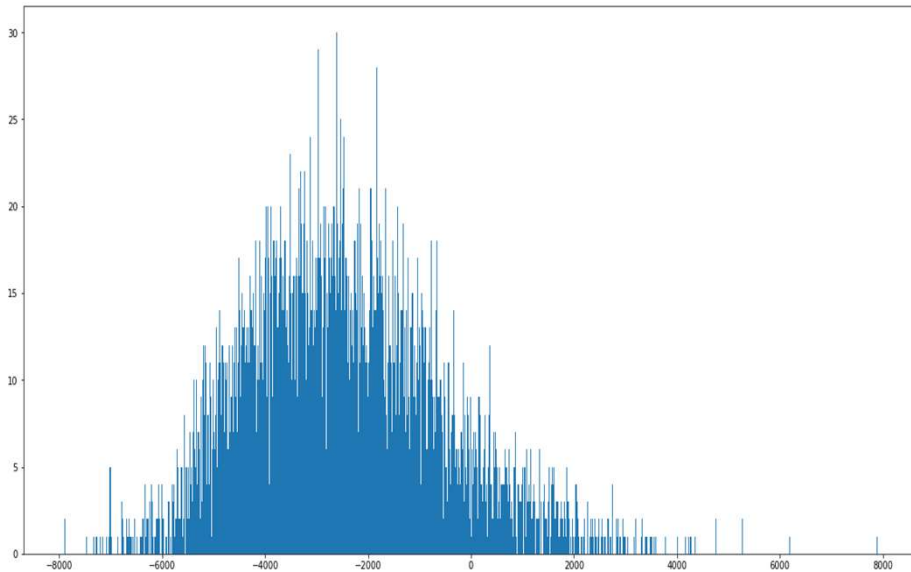
Answers
 $\sim 10^{67}$

Perfect Embedding in Pegasus

Perfect Pegasus Embedding and DQM

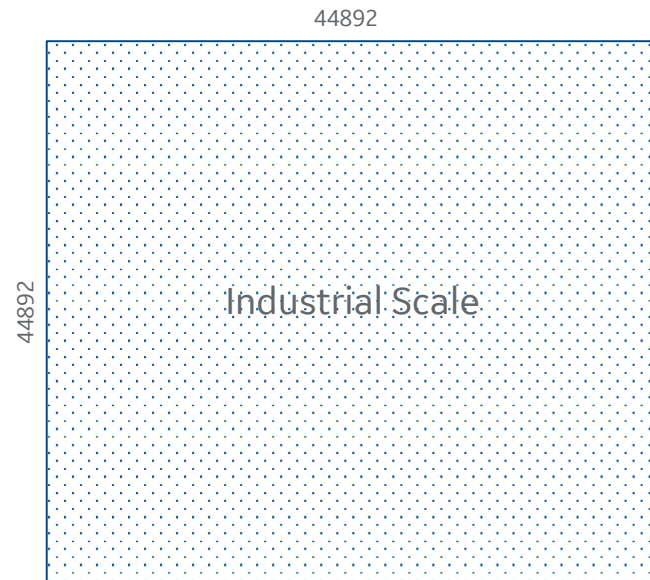
Perfect Embedding- scaling to the qubit limit:

~2615 Repairs with 2 repairs path
Sharing over 31,000 resources
Constrained answers: $\sim 10^{787}$



DQM- scaling beyond the qubit limit:

~15,000 Repairs with ~3 repairs path
Sharing over 6000 resources
Constrained answers: $\sim 10^{6500}$



The answer ->



Challenge: How to motivate change

Mechanical Calculator



Electrical Calculator



• Digital Computers



- Analog
- Quantum



```
function isPrime(n) {
  if (n < 2) return false;
  for (var a = 2; a <= Math.sqrt(n); a++) {
    if (n % a === 0) return false;
  }
  return true;
}

function isPrimeArray(arr) {
  return arr.map(isPrime);
}

function isPrimeArray2(arr) {
  return arr.map(function(n) {
    return isPrime(n);
  });
}
```



Machine

Algorithm

Application

GPP



GPU



FPGA



Quantum





**Thank you for
your time**

Questions?

