CS 470
Spring 2018

Mike Lam, Professor

Other Architectures

(with an aside on linear algebra)

Aside (P3 related): linear algebra

- Many scientific phenomena can be modeled as matrix operations
  - Differential equations, mesh simulations, view transforms, etc.
  - Very efficient on vector processors (including GPUs)
  - Data decomposition and SIMD parallelism
  - Dense matrices vs. sparse matrices
  - Popular packages: BLAS, LINPACK, LAPACK

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} ln(l_1) \\ ln(l_2) \\ ln(l_3) \\ ln(l_4) \\ ln(l_5) \\ ln(l_6) \\ ln(l_7) \end{bmatrix} = \begin{bmatrix} ln(r_{1,3,4}) \\ ln(r_{1,3,5}) \\ ln(r_{2,6}) \\ ln(r_{2,7}) \end{bmatrix}$$

Dense vs. sparse matrices

- A sparse matrix is one in which most elements are zero
  - Could lead to more load imbalances
  - Can be stored more efficiently, allowing for larger matrices
  - Dense matrix operations no longer work
  - It is a challenge to make sparse operations as efficient as dense operations

$$\begin{pmatrix} 11 & 22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 33 & 44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 55 & 66 & 77 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 88 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 99 \end{pmatrix}$$

HPL benchmark

- HPL: LINPACK-based dense linear algebra benchmark
  - Generates a linear system of equations “Ax = b”
    - Chooses b such that x (answer vector) values are known
  - Distributes dense matrix A in block-cyclic pattern
  - LU factorization (similar to Gaussian elimination)
  - Backward substitution to solve system
  - Error calculation to verify correctness
  - Compare max sustained FLOPS (*floating-point operations per section*)
    - Usually significantly less than theoretical machine peak (Rmax vs Rpeak)
  - Serves as proxy app for target workloads (similar characteristics)
  - Compiled on cluster
    - Located in `/shared/apps/hpl-2.1/bin/Linux_PII_CBLAS`

P3 (OpenMP)

- Similar to HPL benchmark
  1) Random generation of linear system (x should be all 1's)
  2) Gaussian elimination
  3) Backwards substitution (row- or column-oriented)

Non-random example

$$\begin{aligned} 3x + 2y - z &= 1 \\ 2x - 2y + 4z &= -2 \\ -x + \tfrac{1}{2}y - z &= 0 \end{aligned}$$

Original system (Ax = b)

$$\left[\begin{array}{ccc:c} 3.0 & 2.0 & -1.0 & 1.0 \\ 2.0 & -2.0 & 4.0 & -2.0 \\ -1.0 & 0.5 & -1.0 & 0.0 \end{array}\right]$$

Augmented matrix [A | b]

Gaussian elimination

$$\left[\begin{array}{ccc:c} 3.0 & 2.0 & -1.0 & 1.0 \\ 0.0 & -3.3 & 4.7 & -2.7 \\ 0.0 & 0.0 & 0.3 & -0.6 \end{array}\right]$$

Upper triangular system

Backward substitution

$$\left[\begin{array}{ccc:c} 1.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & -2.0 \\ 0.0 & 0.0 & 1.0 & -2.0 \end{array}\right]$$

Solved system

P3 notes

- 2D dense matrices in C
  - Often stored in 1D arrays w/ access via array index arithmetic
  - Trace data access patterns to determine dependencies
  - Your goals: 1) **analyze**, 2) **parallelize** (w/ OpenMP), and 3) **evaluate**
  - Example (matrix multiplication):

```
void multiply_matrices(int *A, int *B, int *R, int n)
{
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            R[i*n+j] = 0;
            for (k = 0; k < n; k++) {
                R[i*n+j] += A[i*n+k] * B[k*n+j];
            }
        }
    }
}
```

P3 notes

- 2D dense matrices in C
  - Often stored in 1D arrays w/ access via array index arithmetic
  - Trace data access patterns to determine dependencies
  - Your goals: 1) **analyze**, 2) **parallelize** (w/ OpenMP), and 3) **evaluate**
  - Example (matrix multiplication):

```
void multiply_matrices(int *A, int *B, int *R, int n)
{
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            R[i*n+j] = 0;
            for (k = 0; k < n; k++) {
                R[i*n+j] += A[i*n+k] * B[k*n+j];
            }
        }
    }
}
```

P3 notes

- 2D dense matrices in C
  - Often stored in 1D arrays w/ access via array index arithmetic
  - Trace data access patterns to determine dependencies
  - Your goals: 1) **analyze**, 2) **parallelize** (w/ OpenMP), and 3) **evaluate**
  - Example (matrix multiplication):

```
void multiply_matrices(int *A, int *B, int *R, int n)
{
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            R[i*n+j] = 0;
            for (k = 0; k < n; k++) {
                R[i*n+j] += A[i*n+k] * B[k*n+j];
            }
        }
    }
}
```

read as `R[i,j]`

More OpenMP examples

- Posted in `/shared/cs470`
  - For-loop scheduling (`omp-sched`)
  - Critical sections and deadlock (`omp-deadlock`)
  - The ‘atomic’ directive (`omp-atomic`)
  - Tasks (`omp-qsort`)
  - Matrix multiplication (`omp-matmult`)

Parallel Systems

- Shared memory (uniform global address space)
  - Primary story: **make faster computers**
  - Programming paradigm: threads
  - Technologies: Pthreads, OpenMP
- Distributed (Non-Uniform Memory Access – *NUMA*)
  - Primary story: **add more computers**
  - Programming paradigm: message passing
  - Technologies: MPI (OpenMPI/MPICH), SLURM

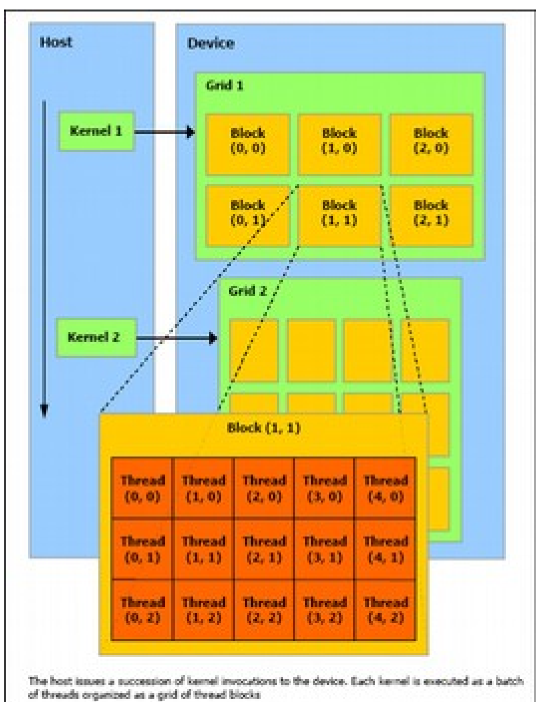
Where do we go from here?

A brief digression into gaming

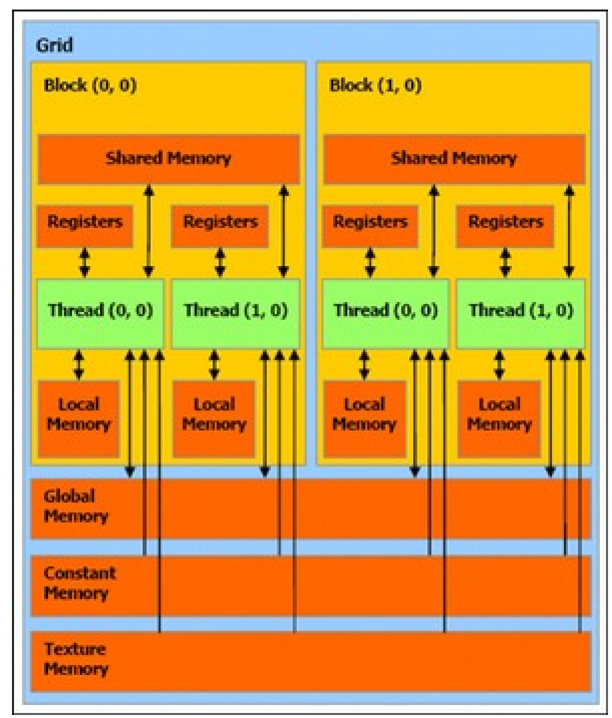
- **1970s**: arcades began using specialized graphics chips
- **1980s**: increasingly sophisticated capabilities (e.g., sprites, blitters, scrolling)
- Early-mid **1990s**: first 3D consoles (e.g., N64) and 3D accelerator cards for PCs
- Late **1990s**: classic wars begin: Nvidia vs. ATI and DirectX vs. OpenGL
- Early **2000s**: new "shaders" enable easier non-graphical use of accelerators
- Late **2000s**: rise of General-Purpose GPU (GPGPU) frameworks
  - 2007: Compute Unified Device Architecture (CUDA) released (newer library: Thrust)
  - 2009: OpenCL standard released
  - 2011: OpenACC standard released
- **2010s**: computation-focused manycore CPUs like Intel Phi (up to 64 cores)

GPU Programming

- "*Kernels*" run on a batch of threads
  - Distributed onto many low-powered GPU cores
  - Grouped into *blocks* of cores and *grids* of blocks
  - Limited instruction set that operates on vector data
  - Must copy data to/from main memory



The host issues a succession of kernel invocations to the device. Each kernel is executed as a batch of threads organized as a grid of thread blocks.



GPU Programming (CUDA)

```
void saxpy_serial(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Low-level control of parallelism on GPU

```
__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n) y[i] = a*x[i] + y[i];
}

// Invoke parallel SAXPY kernel with 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

GPU Programming (CUDA)

```
// Kernel that executes on the CUDA device
__global__ void square_array(float *a, int N)
{
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  if (idx<N) a[idx] = a[idx] * a[idx];
}

// main routine that executes on the host
int main(void)
{
  float *a_h, *a_d;  // Pointer to host & device arrays
  const int N = 10;  // Number of elements in arrays
  size_t size = N * sizeof(float);
  a_h = (float *)malloc(size);        // Allocate array on host
  cudaMalloc((void **) &a_d, size);   // Allocate array on device

  // Initialize host array and copy it to CUDA device
  for (int i=0; i<N; i++) a_h[i] = (float)i;
  cudaMemcpy(a_d, a_h, size, cudaMemcpyHostToDevice);

  // Do calculation on device:
  int block_size = 4;
  int n_blocks = N/block_size + (N%block_size == 0 ? 0:1);
  square_array <<< n_blocks, block_size >>> (a_d, N);

  // Retrieve result from device and store it in host array
  cudaMemcpy(a_h, a_d, sizeof(float)*N, cudaMemcpyDeviceToHost);

  // Print results and cleanup
  for (int i=0; i<N; i++) printf("%d %f\n", i, a_h[i]);
  free(a_h); cudaFree(a_d);
}
```

Must micromanage memory usage and data movement

GPU Programming (OpenACC)

```
#pragma acc data copy(A) create(Anew)
while (error > tol && iter < iter_max)  {
  error = 0.0;

  #pragma acc kernels
  {
    #pragma acc loop
    for (int j = 1; j < n-1; j++) {
      for (int i = 1; i < m-1; i++) {
         Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] +
                              A[j-1][i] + A[j+1][i];
         error = fmax(error, fabs(Anew[j][i] - A[j][i]));
      }
    }

    #pragma acc loop
    for (int j = 1; j < n-1; j++) {
      for (int = i; i < m-1; i++ ) {
        A[j][i] = Anew[j][i];
      }
    }
  }

  if (iter % 100 == 0) printf("%5d, %0.6f\n", iter, error);
  iter++;
}
```

Fewer modifications required; may not parallelize effectively

Hybrid HPC architectures

- Highly parallel on the node
  - Hardware: CPU w/ accelerators
    - GPUs or manycore processors (e.g., Intel Phi and SunWay)
  - Technologies: OpenMP, CUDA, OpenACC, OpenCL
- Distributed between nodes
  - Hardware: interconnect and distributed FS
  - Technologies: MPI, Infiniband, Lustre, HDFS

Top10 systems (Spring 2016)

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
7	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
8	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
9	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834
10	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510

Top10 systems (Spring 2017)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
7	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233

Top10 systems (Spring 2018)

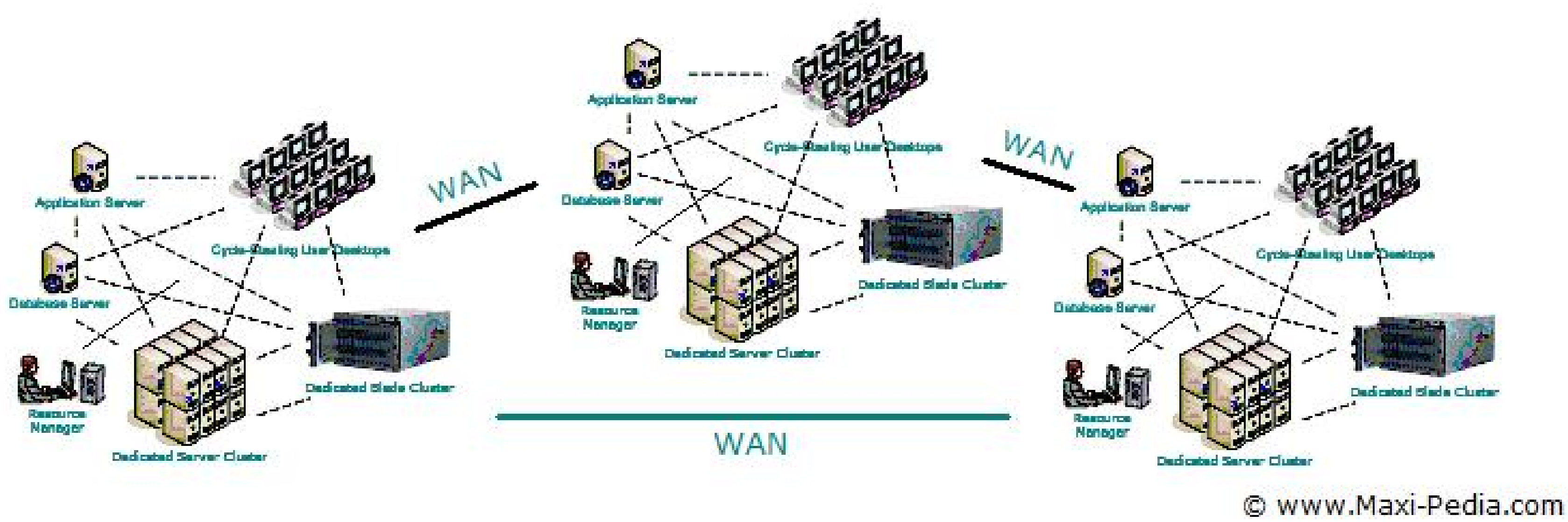
Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
2	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P , NUDT National Super Computer Center in Guangzhou China	3,120,000	33,862.7	54,902.4	17,808
3	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272
4	**Gyoukou** - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz , ExaScaler Japan Agency for Marine-Earth Science and Technology Japan	19,860,000	19,135.8	28,192.0	1,350
5	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc. DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209
6	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , IBM DOE/NNSA/LLNL United States	1,572,864	17,173.2	20,132.7	7,890
7	**Trinity** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,968	14,137.3	43,902.6	3,844
8	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/SC/LBNL/NERSC United States	622,336	14,014.7	27,880.7	3,939
9	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path , Fujitsu Joint Center for Advanced High Performance Computing Japan	556,104	13,554.6	24,913.5	2,719
10	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect , Fujitsu RIKEN Advanced Institute for Computational Science (AICS) Japan	705,024	10,510.0	11,280.4	12,660

Cloud Computing

- Homogenous centralized nodes
  - Infrastructure as a Service (IaaS) and Software as as Service (SaaS)
  - Hardware: large datacenters with thousands of servers and a high-speed internet connection
  - Software: virtualized OS and custom software (Docker, etc.)

Grid Computing

- Heterogenous nodes in disparate physical locations
  - Solving problems or performing tasks of interest to a large number of diverse groups
  - Hardware: different CPUs, GPUs, memory layouts, etc.
  - Software: different OSes, Folding@Home, Condor, GIMPs, etc.



Novel architectures

- Memory-centric
  - Fast memory fabrics w/ in-chip processing
  - Example: HPE The Machine
- Neuromorphic
  - Specialized, low-power hardware that emulates neural networks
  - Example: IBM TrueNorth
- Quantum
  - Leverage quantum superposition and entanglement
  - Example: D-Wave Two and IBM QX