ようこそ (yōkoso) to the colloquium event at

# Computational Systems Biology Lab

# Nara Institute of Science and Technology, Japan

| | |
|---|---|
| **日時(Date)** | 2020/01/14 3:10-4:40 pm (4th period) |
| **場所 (Location)** | エーアイ大講義室, AI,Inc. Seminar Hall （L1） |
| **講演者 (Presenter)** | Prof. Abu Asaduzzaman; Wichita State University, Kansas, USA |
| **題目(Title)** | High Performance Computing, Machine Learning, and Big Data Analytics for Common Good |
| **概要 (Abstract)** | The talk starts with a brief historical background of high-performance computing (HPC), machine learning (ML), and big data (BD) analytics. Today, HPC is essential for modeling and simulation; ML is being used for simulation and data analytics; and BD is vital for acquiring and analyzing data from many different sources. To satisfy tomorrow's computational needs, the convergence of HPC, ML, and BD will be beneficial, if not mandatory. The talk presents three projects developed by the speaker and his team—the first project shows how data and thread regrouping may enhance HPC performance; the second project illustrates a ML-based imaging technique using HPC that can guide real-time surgical procedures; and the third project demonstrates how geospatial BD analytics using HPC and ML can help regional economic success. The talk ends with a discussion on computational challenges where Nara Institute of Science and Technology and Wichita State University can contribute together for common good. |
| **講演言語 (Language)** | English |
| **講演者紹介 (Introduction of Lecturer)** | Abu Asaduzzaman (born 1969, Bangladesh) is an Associate Professor of Computer Engineering at Wichita State University, Kansas, USA. His research interests include high-performance computing, parallel programming, data analytics, and embedded systems. He has authored more than 20 refereed journal articles and more than 95 peer-reviewed conference papers out of his research work. His lab has earned top research designation, a GPU (short for graphics processing unit) Research Center by Nvidia. He has received research grants from Kansas NSF, Nvidia, NetApp, and other organizations. He is a senior member of the IEEE, and member of the ASEE and many student honor societies including Phi Kappa Phi, Tau Beta Pi, Upsilon Pi Epsilon, and Golden Key. As an invited speaker, he has presented his research work at professional forums in Bangladesh, Japan, Sri Lanka, Thailand, Turkey, and USA. Currently he serves as the Director of the Undergraduate (B.S.) Computer Engineering program in his department. |
| **司会(Chair)** | Md. Altaf-Ul-Amin |

**January 14, 2020**

ようこそ (yōkoso) to the colloquium event at

**Computational Systems Biology Lab**

**Nara Institute of Science and Technology, Japan**

# "High Performance Computing, Machine Learning, and Big Data Analytics for Common Good"

*Presenter:*

**Dr. Abu Asaduzzaman (Zaman)**, Associate Professor

Director of CAPPLab; Director of Computer Engineering Programs

Department of Electrical Engineering and Computer Science

**Wichita State University, USA**

**January 14, 2020**

# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## High Performance Computing (HPC)

- **HPC can be a bit hard to define…**
  - 1985, the first National Science Foundation (NSF) HPC partnership among
    (i) the San Diego Supercomputer Center (SDSC) at the University of California San Diego,
    (ii) the Pittsburgh Supercomputer Center (PSC) at the University of Pittsburgh,
    (iii) the Nat’l Center for Supercomputing Apps (NCSA) at the University of Illinois Champagne-Urbana,
    (iv) the Cornell Theory Center (CTC) at Cornell University, and
    (v) the John von Neumann Center (JNC) at Princeton University.

## Machine Learning (ML)

- **ML is a result of advancements in Artificial Intelligence (AI)**
  - 1990s, ML (a subcategory of AI) to give machines the ability to learn

## Big Data (BD) Analytics

BD is huge … 2.5 Quntiliian Bytes (2.3 Trillion Gigabytes)

- **Data analysis is rooted in statistics, pretty long history…**
  - 1990s, data mining, a computational process to discover patterns in large datasets
  - 2010s, Big Data Analysis on the Cloud: Amazon Redshift and Google BigQuery

“High Performance Computing,” https://confluence.xsede.org/pages/viewpage.action?pageId=1677620
“Machine Learning,” https://www.education-ecosystem.com/guides/artificial-intelligence/machine-learning/history/
“A Brief History of Data Analysis,” https://www.flydata.com/blog/a-brief-history-of-data-analysis/
“How Big is Big Data?,” https://www.sisense.com/blog/infographic-big-big-data/

# "High Performance Computing, Machine Learning, and Big Data Analytics for Common Good"

## Outline

- **Introduction**
  - About Myself (Asaduzzaman)
  - Computing Systems: Past, Present, and Future
- **High Performance Computing (HPC)**
  - Hybrid HPC Systems and Parallel Computing/Programming
  - "Regrouping Data/Threads for Improving CPU-GPU Performance"
  - "A Communication-Aware Cache-Controller for HPC Systems"
- **Machine Learning (ML): Medical Image Processing**
  - "Real-Time Image Processing for Breast Cancer Treatment"
- **Geospatial Big Data (BD) Analytics using HPC and ML**
  - "Geospatial Cyberinfrastructure for Common Good"
- **Q/A: Discussion**

**QUESTIONS? Any time, please!**

# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## About Myself

- **Name: A S MD Asaduzzaman, Abu Asaduzzaman (Zaman)**
- **Born: 1969, Bangladesh**
- **Current Affiliation: Wichita State University (WSU), USA**
  - Associate Professor of Computer Engineering, EECS Department
  - Director of CAPPLab and Director of Computer Engineering Programs
- **Scholarly Activities**
  - Grants: Kansas NSF, Nvidia, NetApp, WSU, …
  - Publications: Journal ~ 20, Conference Proceedings ~ 95, …
  - Reviews: NSF, IEEE journals and conferences, …
  - Presentations: Bangladesh, Canada, Japan, Sri Lanka, Thailand, Turkey, and USA

“Computer Architecture and Parallel Programming Laboratory (CAPPLab),” https://www.wichita.edu/academics/engineering/eecs/faculty/Abu/CAPPLab.php

# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## About Myself

- **Dr. Asaduzzaman or Dr. Abu?**
  - ➢ Dr. Zaman (!)
- **Education: PhD from?**
  - ➢ Florida Atlantic University, Boca Raton, Florida, USA
- **Current Affiliation?**
  - ➢ Wichita State University (WSU), Wichita, Kansas, USA

We are not WSU for WASHINGTON STATE UNIVERSITY .

We are  .

WSU WICHITA STATE UNIVERSITY

## Computing Systems: Past, Present, and Future

- **Computing Systems:**
  - … systems that do computations. … computers? …
  - Computations (i.e., information processing) include phenomena ranging from simple calculations to human thinking.
  - … simple/less computations to complex/more computations
  - Need more performance → high performance computing
  - Complex/dynamic computations → machine learning
  - More computations → big data analytics
  - Other issues: security, etc. (beyond the scope of this presentation)

# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## Computing Systems: Past, Present, and Future

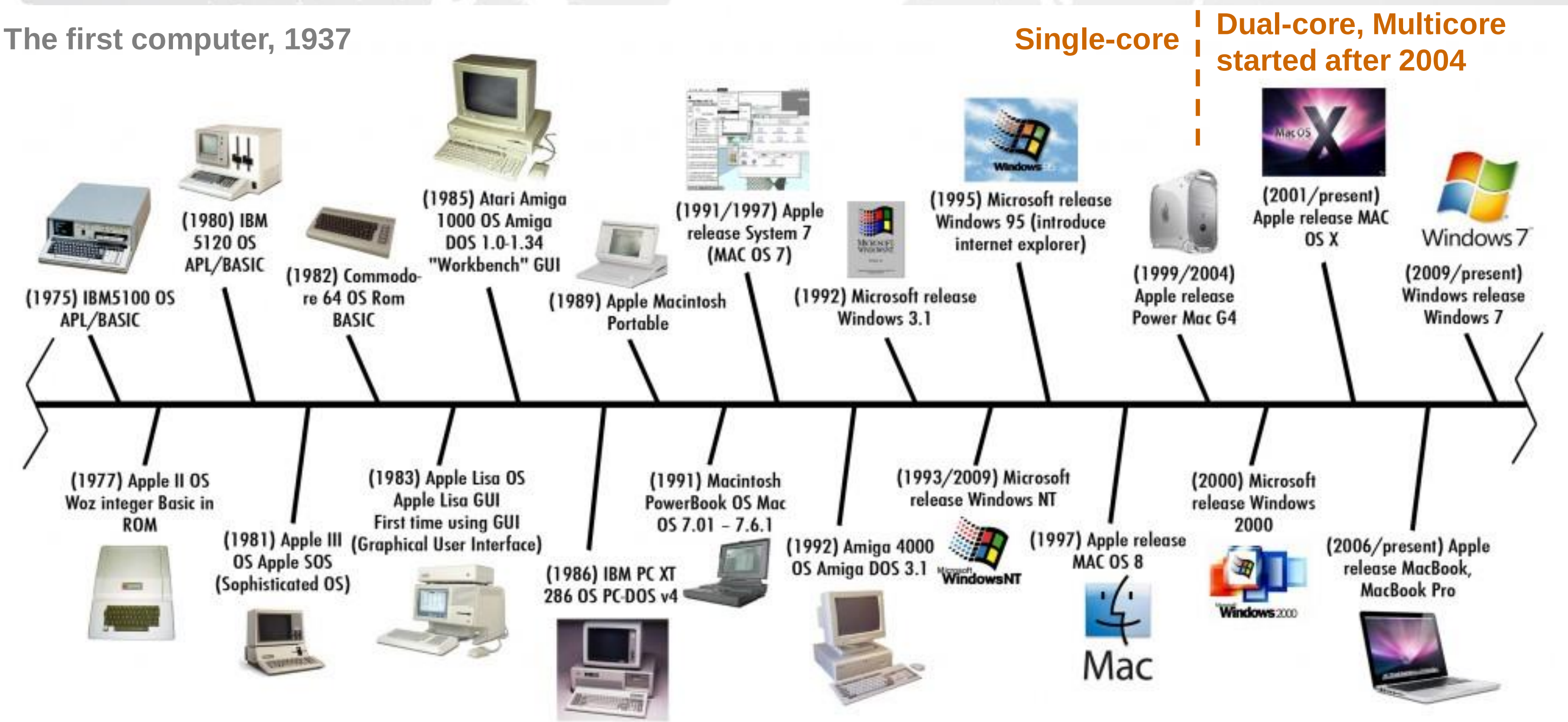
- **Evolution of Computer Systems** What do we see?


“Evolution of products,” https://collaborativebriefresearch.wordpress.com/2011/01/31/evolution-of-products/

# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## Computing Systems: Past, Present, and Future

- **Computer Systems:** 1975–2009




Hybrid High Performance Computing (HPC) Systems, since 1985

Quantum Computing, since before 1980

“Invention/evolution of the relay computer,” https://prezi.com/d_-cpssu6zyx/inventionevolution-of-the-relay-computer/

# Computing Systems: Past, Present, and Future

- **Timeline of Programming Languages** When started?

| Year | Name | Contributor(s) | Predecessor(s) |
|---|---|---|---|
| 1804 | Jacquard Loom | Joseph M. Jacquard | None (unique language) |
| 1946 | ENIAC Short Code | R. Clippinger, J. von Neumann, A. Turing | ENIAC coding system |
| 1947 | ARC Assembly | Kathleen Booth | ENIAC coding system |
| 1956 | LISP (concept) | John McCarthy | IPL |
| 1956 | Fortran I … IV | J.W. Backus at IBM | FORTRAN |
| 1959 | COBOL (concept) | The CODASYL Committee | FLOW-MATIC, COMTRAN, FACT |
| 1964 | BASIC | J.G. Kemeny & T.E. Kurtz at Dartmouth College | FORTRAN II, JOSS |

"Timeline of programming languages," https://en.wikipedia.org/wiki/Timeline_of_programming_languages

# Computing Systems: Past, Present, and Future

- **Timeline of Programming Languages** Python or Java?

| Year | Name | Contributor(s) | Predecessor(s) |
|---|---|---|---|
| 1970 | Pascal | N. Wirth and K. Jensen | ALGOL 60, ALGOL W |
| 1972 | (K&R) **C** | Dennis Ritchie | B, BCPL, ALGOL, 68 |
| 1972 | Prolog / SQL | A. Colmerauer / IBM | 2-level W-Grammar / ALPHA, Quel (Ingres) |
| 1978 | MATLAB (?) | C. Moler, U. New Mexico | Fortran |
| 1980 | Ada 80 | J. Ichbiah, C Honeywell B | Green |
| 1983 | **C++** | Bjame Stroustrup | C with Classes |
| 1989 | Python | Guido van Rossum | ABC, SETL |
| 1995 | Java | J. Gosling, Sun Microsys | C, Simula 67, C++, Smalltalk, Ada 83, Objective-C, Mesa |

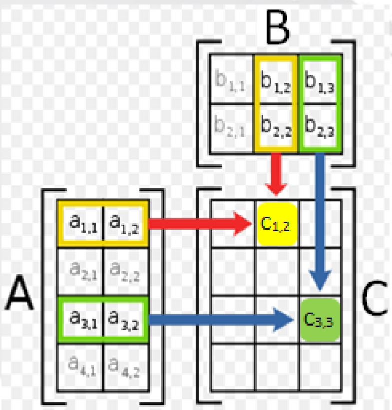
“Timeline of programming languages,” https://en.wikipedia.org/wiki/Timeline_of_programming_languages

# High Performance Computing:
# Thinking/Programming in Parallel

## Matrix Multiplication

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}\begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

- **[C] = [A] [B]**
  - **2 x 2 Matrix**
  - **8 (i.e., 2 * 2^2) multiplications**
  - **4 (i.e., 1 * 2^2) additions**
  - **Real example:**




$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 26 & 30 \\ 38 & 44 \end{bmatrix}$$

$A_{1,1} = 1; A_{1,2} = 3 \qquad B_{1,1} = 5; B_{1,2} = 6$

$A_{2,1} = 2; A_{2,2} = 4 \qquad B_{2,1} = 7; B_{2,2} = 8$

$C_{1,1} = 1x5 + 3x7 = 5 + 21 = 26$

$C_{1,2} = 1x6 + 3x8 = 6 + 24 = 30$

$C_{2,1} = 2x5 + 4x7 = 10 + 28 = 38$

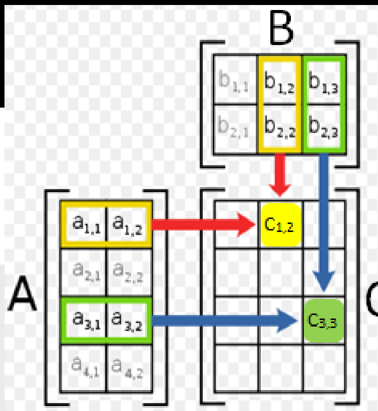
$C_{2,2} = 2x6 + 4x8 = 12 + 32 = 44$

$$C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$$

$$C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$$

$$C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$$

$$C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$$




# Matrix Multiplication (+)

■[C] = [A] [B]

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{pmatrix} * \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} \\ c_{1,0} & c_{1,1} & c_{1,2} \\ c_{2,0} & c_{2,1} & c_{2,2} \end{pmatrix}$$

$c_{0,0} = a_{0,0}b_{0,0} + a_{0,1}b_{1,0} + a_{0,2}b_{2,0}$ $c_{0,1} = a_{0,0}b_{0,1} + a_{0,1}b_{1,1} + a_{0,2}b_{2,1}$ $c_{0,2} = a_{0,0}b_{0,2} + a_{0,1}b_{1,2} + a_{0,2}b_{2,2}$

$c_{1,0} = a_{1,0}b_{0,0} + a_{1,1}b_{1,0} + a_{1,2}b_{2,0}$ $c_{1,1} = a_{1,0}b_{0,1} + a_{1,1}b_{1,1} + a_{1,2}b_{2,1}$ $c_{1,2} = a_{1,0}b_{0,2} + a_{1,1}b_{1,2} + a_{1,2}b_{2,2}$

$c_{2,0} = a_{2,0}b_{0,0} + a_{2,1}b_{1,0} + a_{2,2}b_{2,0}$ $c_{2,1} = a_{2,0}b_{0,1} + a_{2,1}b_{1,1} + a_{2,2}b_{2,1}$ $c_{2,2} = a_{2,0}b_{0,2} + a_{2,1}b_{1,2} + a_{2,2}b_{2,2}$

- **3 x 3 Matrix**
- **How many multiplications and additions?**
- **27 (i.e., 3 * 3^2 i.e., 3^3) multiplications**
- **18 (i.e., 2 * 3^2 i.e., (3 – 1) * 3^2) additions**

# Programming (Traditional C, Pthread/C, CUDA/C)

- **Matrix Multiplication (in C)**

```
void matrixMul()
{
  int i, j, k;
  int collector;
  for (i=0; i<matstr.x ; i++) {
    for (j=0;j<matstr.y;j++) {
      for (k=0; k<matstr.x; k++) {
        collector += (matstr.b[i*matstr.x + k] * matstr.a
[matstr.y*k + j]);
      }
      matstr.c[i*matstr.x + j] = collector;
      collector = 0;
    }
  }
}/* end matrixMul */
```

## Programming (Traditional C, Pthread/C, CUDA/C)

### ■ Matrix Multiplication (in Pthread/C)

```
int main (int argc, char *argv[])
{
  ...
  pthread_attr_t attr;
  ...
  pthread_attr_init(&attr);
  pthread_attr_setdetachstate(&attr
  ...
  for(i=0;i<NUMTHRDS;i++)
    pthread_create(&callThd[i], &at
  pthread_attr_destroy(&attr); /* b
  for(i=0;i<NUMTHRDS;i++)
    pthread_join(callThd[i], &statu
  pthread_mutex_destroy(&mutexsum);
  ...
  pthread_exit(NULL);
  return 0;
}/* end main */
```

```
void *matrixMul(void *arg)
{
  int i, j, start, end, len ;
  int row, col;
  long offset;
  offset = (long)arg;
  len = matstr.len;
  start = offset*len;
  end = start + len;

  for (i=start; i<end ; i++) {
    row = (i==0? 0: i/matstr.x);
    col = (i==0? 0: i%matstr.y);
    for (j=0; j<matstr.x; j++) {
      matstr.c[col*matstr.x + row] += (matstr.b[col*matstr.x +
j] * matstr.a[matstr.y*j + row]);
    }
    /* printf("======TID %d WORKING ON %d %d==========\n",
(int)arg, col, row); */
  }
  pthread_exit((void*) 0);
}/* end *matrixMul */
```

Asaduzzaman, A., Sibai, F.N., and Elsayed, H.; "Performance and Power Comparisons of MPI vs Pthread Implementations on Multicore Systems," in 2013 International Conference on Innovations in Information Technology (ITT'13), Al Ain, UAE, Mar. 17-19, 2013.

# Matrix Multiplication (+)

- **[C] = [A] [B]**

  - **[C] = [A] [B] =** $\begin{bmatrix} 3 & 4 & 1 & 6 \\ 1 & 2 & 5 & 7 \\ 5 & 1 & 2 & 9 \\ 4 & 3 & 5 & 6 \end{bmatrix} \begin{bmatrix} 5 & 6 & 9 & 3 \\ 4 & 5 & 3 & 1 \\ 1 & 1 & 8 & 4 \\ 3 & 1 & 4 & 1 \end{bmatrix}$
  - **4 x 4 Matrix**
  - **How many multiplications and additions?**
  - **64 (i.e., 4^3) multiplications → N^3 multiplications**
  - **48 (i.e., 3 * 4^2) additions → (N – 1)N^2 additions**

# Matrix Multiplication (+)

- **Divide 4x4 matrix into four 2x2 matrices**
  - **4 x 4 Matrix**
  - **64 (i.e., 4^3) multiplications**
  - **48 (i.e., 3 * 4^2) additions**

  - **2 x 2 Matrix**
  - **8 (i.e., 2^3) multiplications**
  - **4 (i.e., 1 * 2^2) additions**

  - **Are we reducing *s/+s?**
  - **What is the message?**

$$\begin{array}{cc} & \begin{array}{cc} A_{1,1} & A_{1,2} \end{array} \\ \mathbf{A} & \begin{bmatrix} 3 & 4 & 1 & 6 \\ 1 & 2 & 5 & 7 \\ 5 & 1 & 2 & 9 \\ 4 & 3 & 5 & 6 \end{bmatrix} \end{array} \begin{array}{c} \begin{bmatrix} 5 & 6 & 9 & 3 \\ 4 & 5 & 3 & 1 \\ 1 & 1 & 8 & 4 \\ 3 & 1 & 4 & 1 \end{bmatrix} \end{array} \mathbf{B}$$

$$A_{1,1} = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \quad A_{1,2} = \begin{bmatrix} 1 & 6 \\ 5 & 7 \end{bmatrix} \quad B_{1,1} = \begin{bmatrix} 5 & 6 \\ 4 & 5 \end{bmatrix} \quad B_{1,2} = \begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix}$$

$$A_{2,1} = \begin{bmatrix} 5 & 1 \\ 4 & 3 \end{bmatrix} \quad A_{2,2} = \begin{bmatrix} 2 & 9 \\ 5 & 6 \end{bmatrix} \quad B_{2,1} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \end{bmatrix} \quad B_{2,2} = \begin{bmatrix} 8 & 4 \\ 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

## Matrix Multiplication (+)

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}\begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

■**Divide 4x4 matrix into four 2x2 matrices**

- **[C] = [A] [B]**

$$A_{1,1} = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \quad A_{1,2} = \begin{bmatrix} 1 & 6 \\ 5 & 7 \end{bmatrix} \quad B_{1,1} = \begin{bmatrix} 5 & 6 \\ 4 & 5 \end{bmatrix} \quad B_{1,2} = \begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix}$$

$$A_{2,1} = \begin{bmatrix} 5 & 1 \\ 4 & 3 \end{bmatrix} \quad A_{2,2} = \begin{bmatrix} 2 & 9 \\ 5 & 6 \end{bmatrix} \quad B_{2,1} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \end{bmatrix} \quad B_{2,2} = \begin{bmatrix} 8 & 4 \\ 4 & 1 \end{bmatrix}$$

- **Say, we have unlimited 2 x 2 Matrix solvers with 8 MULT**
- **Then it takes "only" 2 * 8 MULT time unit**
- **Do we have unlimited solvers/cores?**

$$C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$$

$$C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$$

$$C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$$

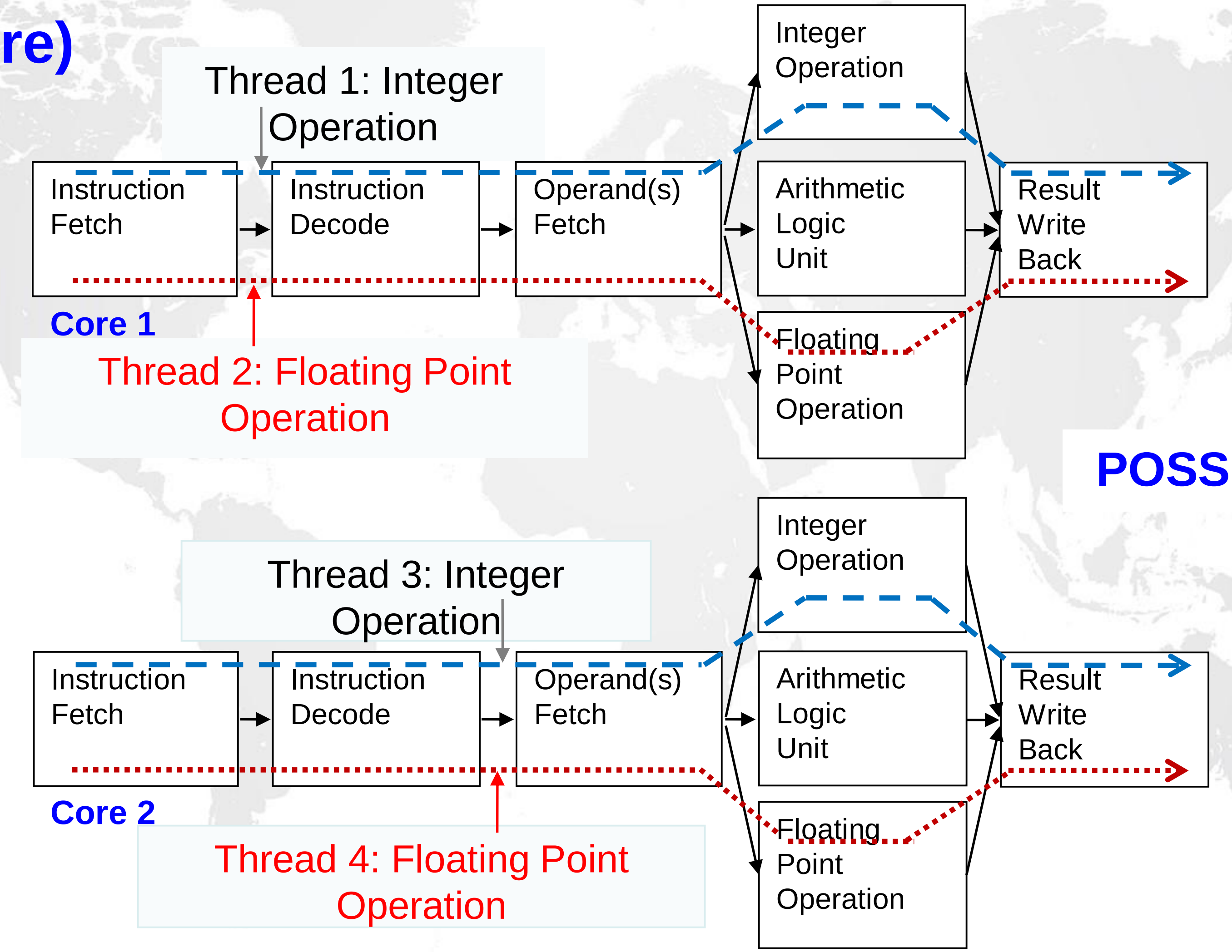
$$C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$$

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}\begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

# **High Performance Computing:** Thinking/Programming in Parallel

## Threads 1, 2, 3, and 4: INT & FP Operations

**(Multicore)**




**POSSIBLE?**

# Programming (Traditional C, Pthread/C, CUDA/C)

## ■ Matrix Multiplication (in CUDA/C)

```
int main( void ) {
   cudaEvent_t start, stop;
   float   *a, *b, c, *partial_c;
   float   *dev_a, *dev_b, *dev_partial_c;

   printf( "N=%d, blocksPerGrid=%d\n", N, blocksPerGrid);

   // allocate memory on the cpu side
   a = (float*)malloc( N*sizeof(float) );
   b = (float*)malloc( N*sizeof(float) );
   partial_c = (float*)malloc( blocksPerGrid*sizeof(float) );

   HANDLE_ERROR( cudaEventCreate( &start ) );
   HANDLE_ERROR( cudaEventCreate( &stop ) );
   // capture the start time
   HANDLE_ERROR( cudaEventRecord( start, 0 ) );
   printf( "ElapsedTime: 0.0 ms \n" );
   // allocate the memory on the GPU
   HANDLE_ERROR( cudaMalloc( (void**)&dev_a,
                          N*sizeof(float) ) );
   HANDLE_ERROR( cudaMalloc( (void**)&dev_b,
                          N*sizeof(float) ) );
   HANDLE_ERROR( cudaMalloc( (void**)&dev_partial_c,
                          blocksPerGrid*sizeof(float) ) );

   // fill in the host memory with data
   for (int i=0; i<N; i++) {
       a[i] = i;
       b[i] = i*2;
   }

   // copy the arrays 'a' and 'b' to the GPU
   HANDLE_ERROR( cudaMemcpy( dev_a, a, N*sizeof(float),
                          cudaMemcpyHostToDevice ) );
   HANDLE_ERROR( cudaMemcpy( dev_b, b, N*sizeof(float),
                          cudaMemcpyHostToDevice ) );

   dot<<<blocksPerGrid,threadsPerBlock>>>( dev_a, dev_b,
                                        dev_partial_c );
```

```
PU to the CPU
dev_partial_c,
id*sizeof(float),
eviceToHost ) );

ing results
0 ) );
top ) );

elapsedTime, start, stop ) );
edTime );
t ) );
 ) );

+1)/6)
uares(4));
quares( (float)(N - 1) ) );

 ) );

   free( partial_c );
}
```

```
__global__ void dot( float *a, float *b, float *c ) {
    __shared__ float cache[threadsPerBlock];
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    int cacheIndex = threadIdx.x;

    float   temp = 0;
    while (tid < N) {
        temp += a[tid] * b[tid];
        tid += blockDim.x * gridDim.x;
    }

    // set the cache values
    cache[cacheIndex] = temp;

    // synchronize threads in this block
    __syncthreads();

    // for reductions, threadsPerBlock must be a power of 2
    // because of the following code
    int i = blockDim.x/2;
    while (i != 0) {
        if (cacheIndex < i)
            cache[cacheIndex] += cache[cacheIndex + i];
        __syncthreads();
        i /= 2;
    }

    if (cacheIndex == 0)
        c[blockIdx.x] = cache[0];
}
```

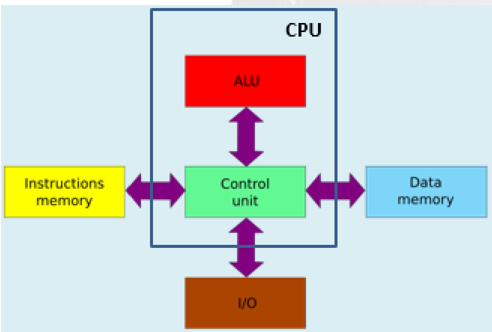
# Outline

QUESTIONS? Any time, please!

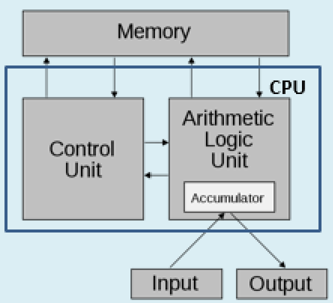
- **Introduction**
  - About Myself (Asaduzzaman)
  - Computing Systems: Past, Present, and Future
- **High Performance Computing (HPC)**
  - Hybrid HPC Systems and Parallel Computing/Programming
  - “Regrouping Data/Threads for Improving CPU-GPU Performance”
  - “A Communication-Aware Cache-Controller for HPC Systems”
- **Machine Learning (ML): Medical Image Processing**
  - “Real-Time Image Processing for Breast Cancer Treatment”
- **Geospatial Big Data (BD) Analytics using HPC and ML**
  - “Geospatial Cyberinfrastructure for Common Good”
- **Q/A: Discussion**

# High Performance Computing:
## HPC Systems and Parallel Computing

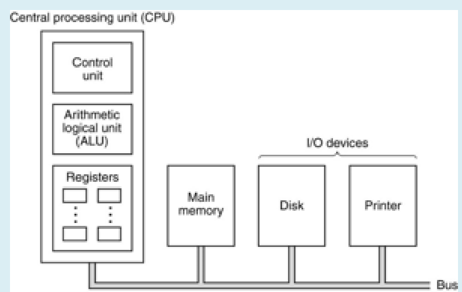
CPU – Central Processing Unit
GPU – Graphics Processing Unit
SMT – Simultaneous Multi-Threading



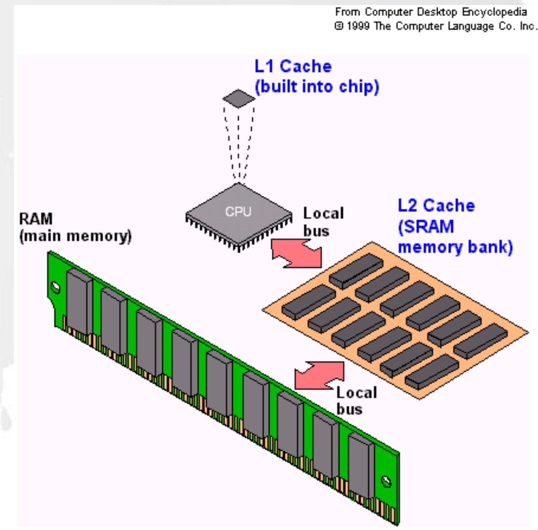

Harvard Arch
1947




Von Neumann Arch
1945-1951




PDP-8 Arch
1965

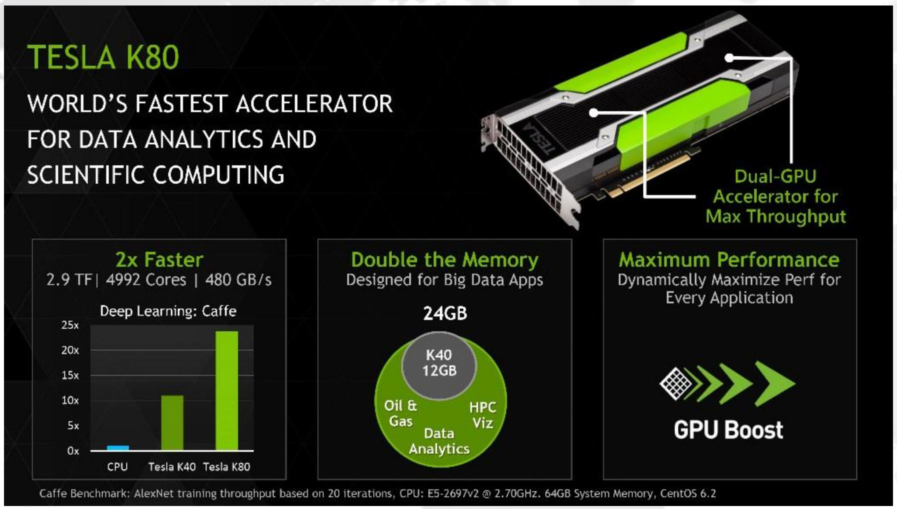



A Computer System

Name of the Game: performance, power, price, …

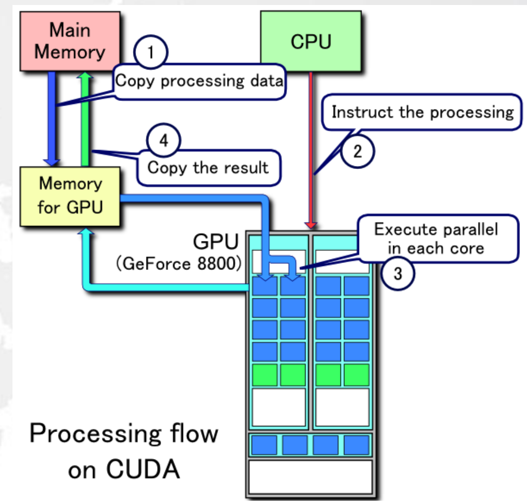
# High Performance Computing:
# HPC Systems and Parallel Computing

CPU – Central Processing Unit
GPU – Graphics Processing Unit
SMT – Simultaneous Multi-Threading




Compute Unified Device Architecture (CUDA) – a parallel computing platform and application programming interface (API) model created by Nvidia

## Parallel Programming:
OpenMP, Open MPI, GPU Programming




**CPU-GPU System**

**Parallel Computing: things to remember**

One woman can make a baby in 9 months.
Can 9 woman make a baby in 1 month? No
But 9 women can make 9 babies in 9 months.

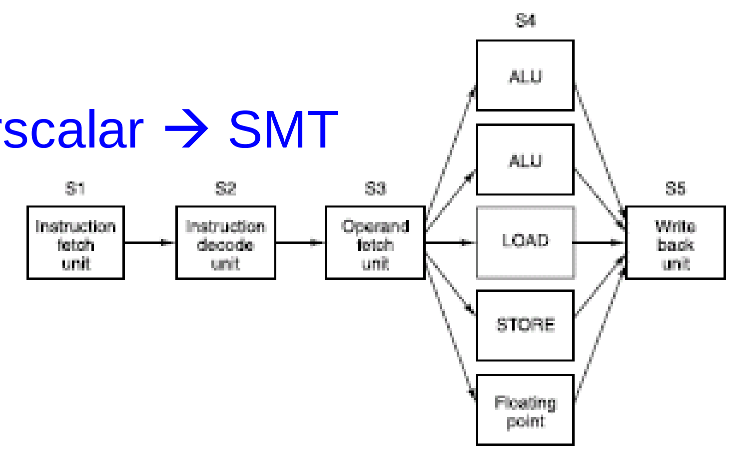
# High Performance Computing:
## HPC Systems and Parallel Computing

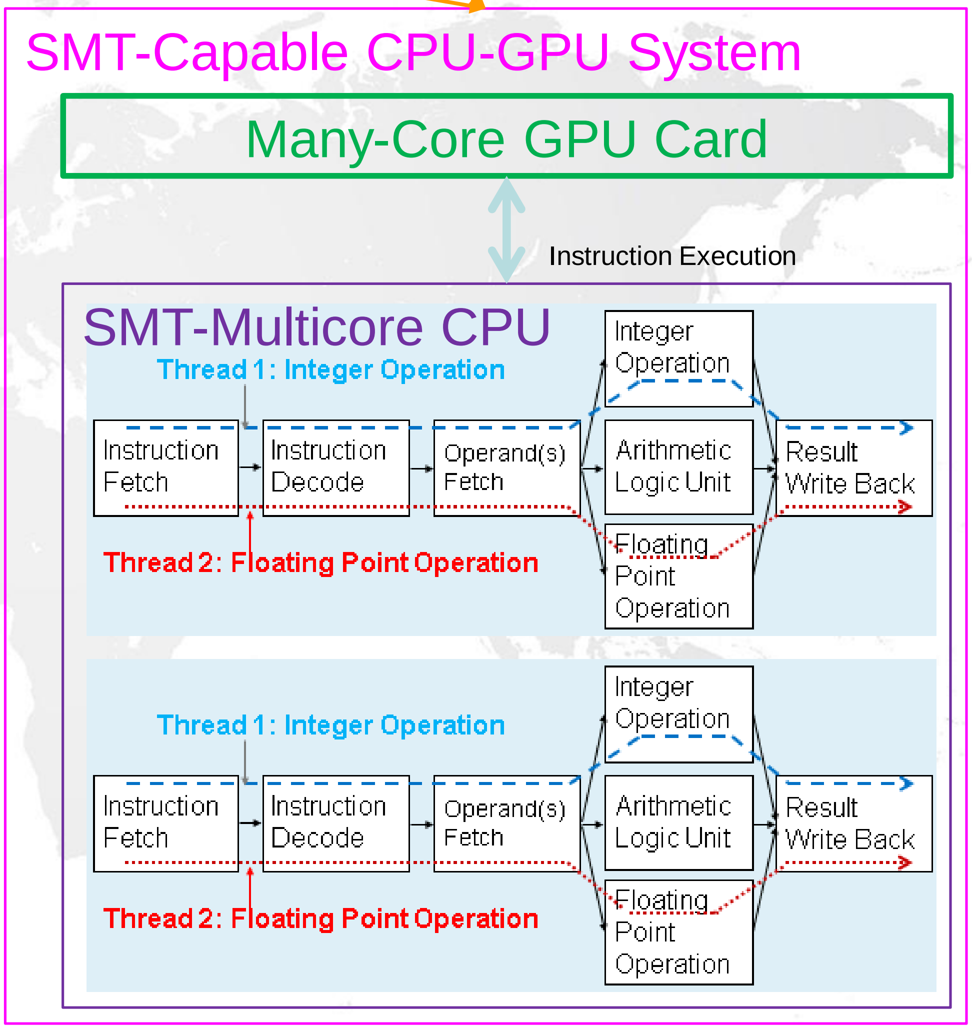
CPU – Central Processing Unit
GPU – Graphics Processing Unit
SMT – Simultaneous Multi-Threading

A process is a running program. A process can generate many processes (called threads). …

Pipelining → Instruction-Level Parallelisms (ILP)
→ Thread-Level Parallelisms (TLP)


Superscalar → SMT




"Intel Xeon Phi has up to 72 cores," http://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors.html
"Nvidia Tesla K80 has up to 4992 cores," http://www.nvidia.com/object/tesla-k80.html

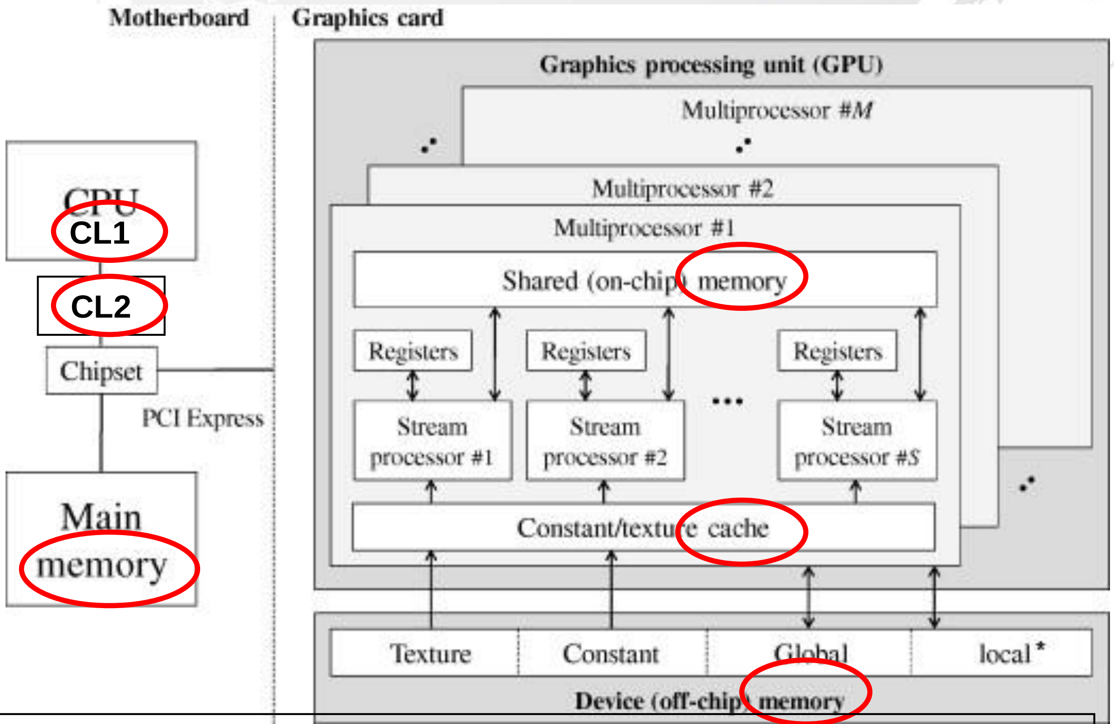
# High Performance Computing:
## HPC Systems and Parallel Computing

CPU – Central Processing Unit
GPU – Graphics Processing Unit
SMT – Simultaneous Multi-Threading

*SMT-Capable CPU-GPU Systems →*
*High-Performance Computer (HPC) Systems*
*support parallel computing*

CL1 – Level-1 Cache
CL2 – Level-2 Cache
Cache and memory are very power-hungry. *More energy consumption, more heat dissipation!*

*HPC: CPU i7-980X 130W,*
*GPU Tesla K80 300W [1, 2]*
*Tianhe-1A consumes 4.04 MW;*
*for 4 MW at $0.10/kWh*
*is $400 an hour or*
*about $3.5 million per year. [3]*




**Name of the Game: performance, power, price, …**

[1] https://ark.intel.com/products/series/79666/Legacy-Intel-Core-Processors
[2] https://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf
[3] https://en.wikipedia.org/wiki/Supercomputer

# High Performance Computing:
# HPC Systems and Parallel Computing

CPU – Central Processing Unit
GPU – Graphics Processing Unit
SMT – Simultaneous Multi-Threading

## ❑ HPC Systems

- If SMT-capable 16-core CPU and 5000-core GPU card are used to build a HPC system, it offers about 9 Tera (10^12) FLOPS and costs about $5K. [1]
- *HPC: CPU i7-980X 130W, GPU Tesla K80 300W*

## ❑ Supercomputers

- A supercomputer may have more or less 300,000 processing cores and operate at Peta (10^15) FLOPS; however, it costs tens of millions of dollars. [2, 3]
- *Tianhe-1A: 4.04 MW (about $3.5 million per year)*

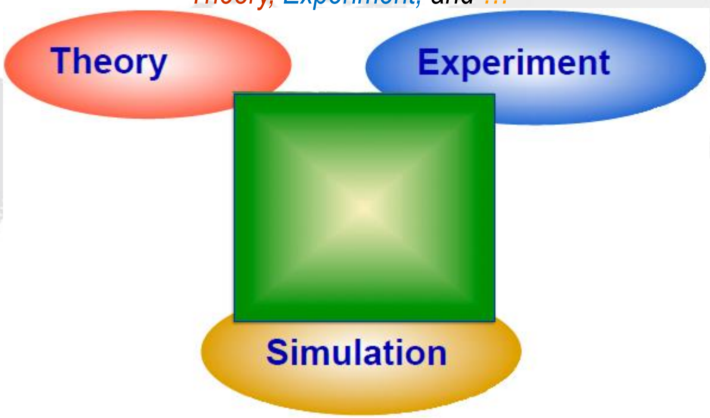
**Name of the Game: performance, power, price, …**

[1] https://insidehpc.com/hpc-basic-training/what-is-hpc/
[2] https://en.wikipedia.org/wiki/Supercomputer
[3] https://www.anandtech.com/show/8729/nvidia-launches-tesla-k80-gk210-gpu

# High Performance Computing:
## Applications of HPC

### The "Third Pillar" of Science?

*Theory, Experiment, and …*

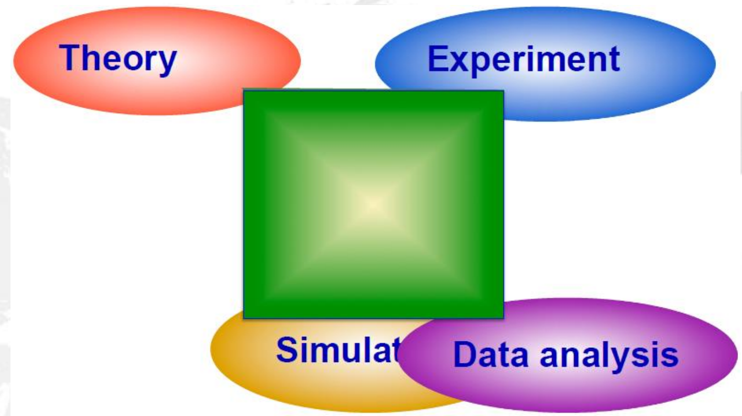



HPC Simulation to understand things that are:
too big, too small, too fast, too slow, too expensive, or too dangerous
for experiments

the universe

proteins and diseases

### The "4th Paradigm" of Science?




Data analytics to analyze data sets:
too big, too complex, too fast (streaming), too noisy, or too heterogeneous
for theory alone

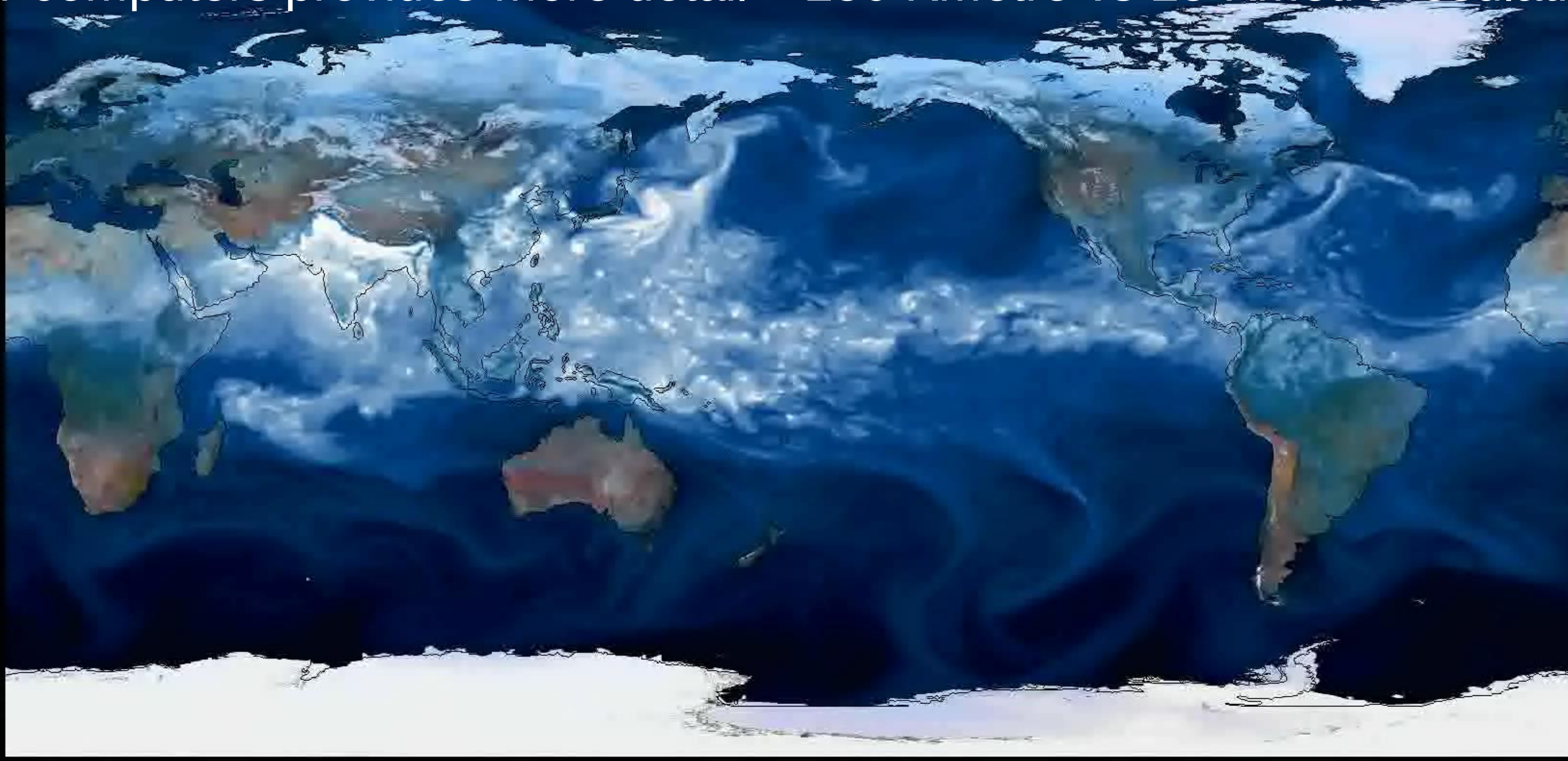
images from telescopes

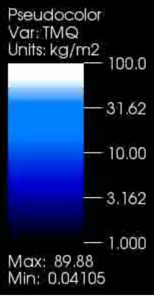
genomes from sequencers

https://www.youtube.com/watch?v=o7DQd8FkA6M&feature=youtu.be

# **High Performance Computing:**
# Applications of HPC

Simulations Show the Effects of Climate Change in Hurricanes:
Faster computers provides more detail – 250 Kmetre vs 25 Kmetre resulation






Michael Wehner, Prabhat, Chris Algieri, Fuyu Li, Bill Collins, Lawrence Berkeley National Laboratory;
Kevin Reed, University of Michigan;
Andrew GeOelman, Julio Bacmeister, Richard Neale, National Center for Atmospheric Research

https://www.youtube.com/watch?v=o7DQd8FkA6M&feature=youtu.be

# **High Performance Computing:**
# A Poisson Solver

## Poisson’s Equation

- … to solve electrostatic problems the Poisson's Equation (with Laplacian operator) can be used.

$$\nabla^2 \varphi = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = -\frac{\rho}{\epsilon}$$

where, φ is electric potential, ρ is the total volume charge density, and ε is permittivity of the medium.

- If the charge density is zero all over the region, the Poison's Equation becomes Laplace's equation:

$$\nabla^2 \varphi = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0$$

Wu, D. and Chen, J., “Efficient characterizations of composite materials electrical properties based on GPU accelerated finite difference method,” in IEEE Antennas and Propagation Society International Symposium, Toronto, Canada, 2010.

# High Performance Computing:
# A Poisson Solver

## Poisson's Equation → Laplace's Equation

- For very uniform material, Laplace's equation can be considered as a three-dimensional steady state heat equation as shown below and can be solved using the discrete approach by writing computer program.

$$(\varphi_{i+1,j,k} - \varphi_{i,j,k})/dx + (\varphi_{i,j+1,k} - \varphi_{i,j,k})/dy + (\varphi_{i,j,k+1} - \varphi_{i,j,k})/dz + (\varphi_{i,j,k} - \varphi_{i-1,j,k})/dx + (\varphi_{i,j,k} - \varphi_{i,j-1,k})/dy + (\varphi_{i,j,k} - \varphi_{i,j,k-1})/dz = 0$$

- Programs: Serial Vs Parallel
- Parallel: OpenMP, Open MPI, GPU/CUDA (shared memory)

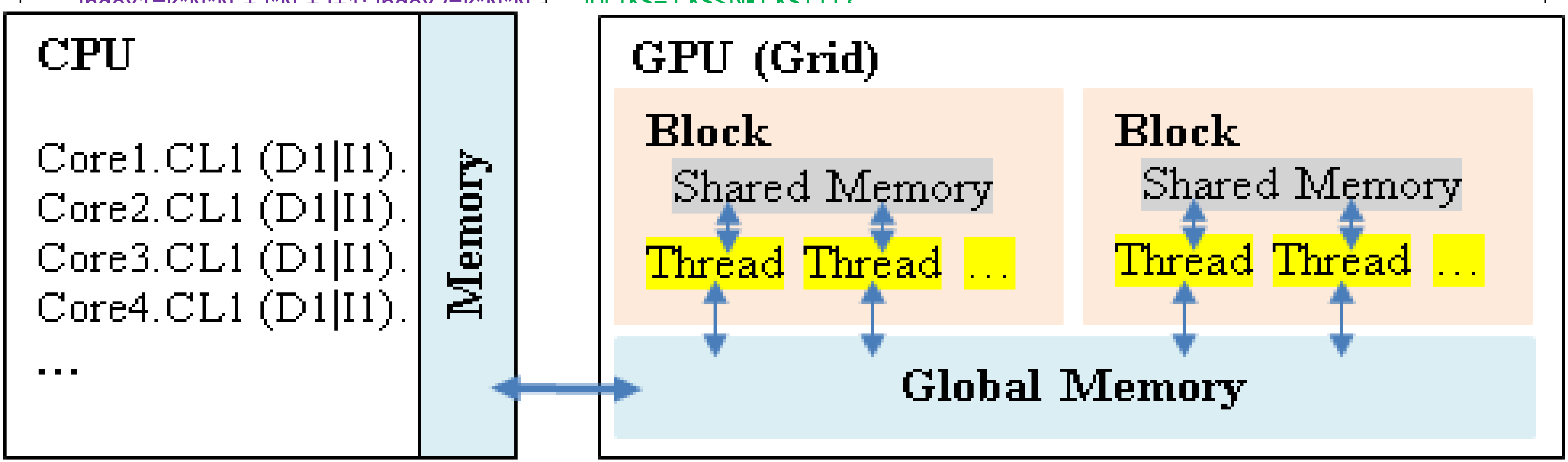
Wu, D. and Chen, J., "Efficient characterizations of composite materials electrical properties based on GPU accelerated finite difference method," in IEEE Antennas and Propagation Society International Symposium, Toronto, Canada, 2010.

# High Performance Computing:
# A Poisson Solver

## Laplace Equation: CUDA Code without and with GPU Shared Memory

```
/* CUDA/GPGPU implementation of the heat t
__global__ void Heat_Transfer_GPU(float *A,
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int j = blockIdx.y * blockDim.y + threadIdx.y;
  int k, index, index1, index2, index3, index4,
  for (k=1;k<N-1;k++) {
    index = k*N*N + j*N + i;
    index1=k*N*N + j*N + i+1; index2=k*N*N
```

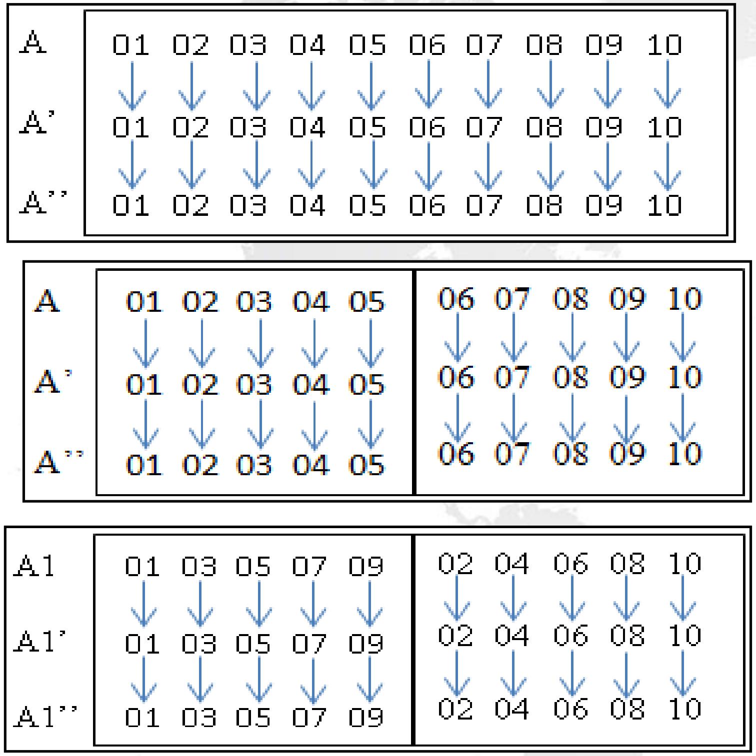
```
/* CUDA/GPGPU implementation of the heat transfer equation with shared memory */
__global__ void Heat_Transfer_GPU_SM(float *A, float *B, int N) {
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int j = blockIdx.y * blockDim.y + threadIdx.y;
  int is = threadIdx.x ; int js = threadIdx.y;
  __shared__ float As[THRDIM][THRDIM];
  int ks, index, index1, index2, index3, index4, index5, index6;
  As[threadIdx.x][threadIdx.y] = A[index];
   __syncthreads();
  for (ks=1;ks<N-1;ks++) {
```



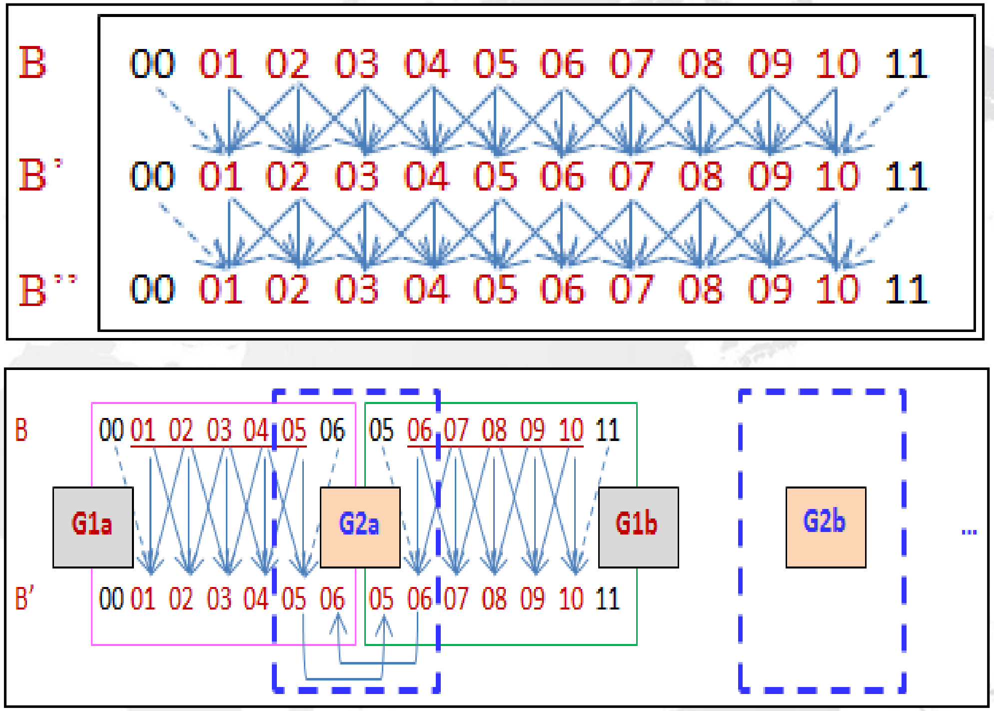

Asaduzzaman, A., Yip, C.M.*, Kumar, S., and Asmatulu, R.; “Fast, Effective, and Adaptable Computer Modeling and Simulation of Lightning Strike Protection on Composite Materials,” in IEEE SoutheastCon Conference 2013, Jacksonville, Florida, April 4-7, 2013.

# High Performance Computing:
## "Regrouping Data/Threads for Improving CPU-GPU Performance"

## Data/Task Partitioning/Regrouping




Processing data without independency



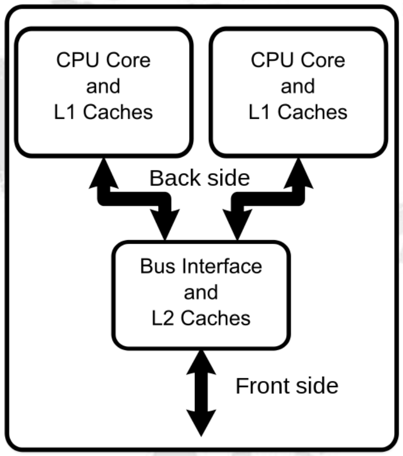

Processing data with dependency

Problems include

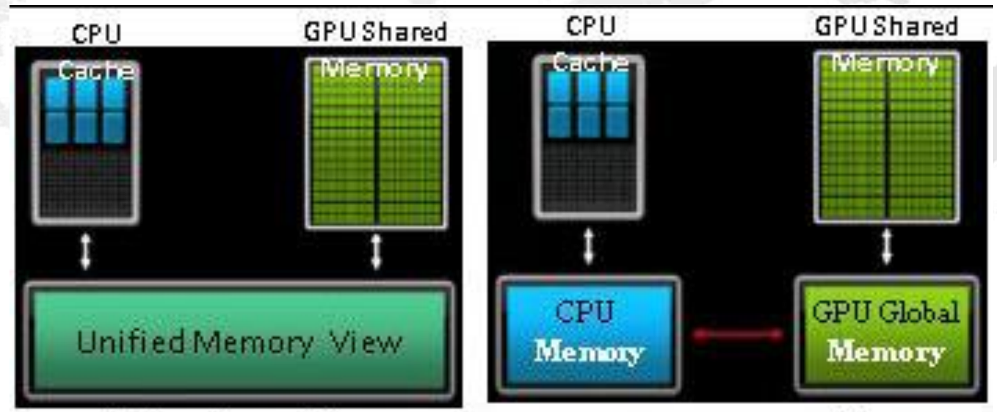
- Synchronization → performance, errors, …

Asaduzzaman, A., Gummadi, D., and Yip, C.M., "A Talented CPU-to-GPU Memory Mapping Technique," in IEEE SoutheastCon 2014, Lexington, Kentucky, March 13-16, 2014.

# CPU-GPU System and Workflow

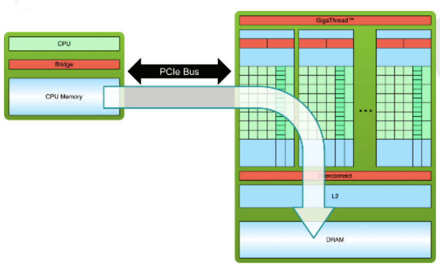



A multicore CPU with two levels of caches

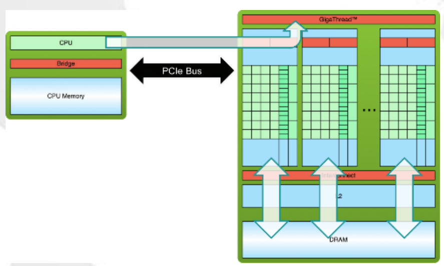



(a) Developer View (b) Actual CPU GPU Memory

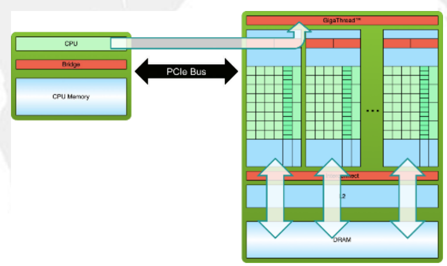
A multicore CPU with many-core GPU system



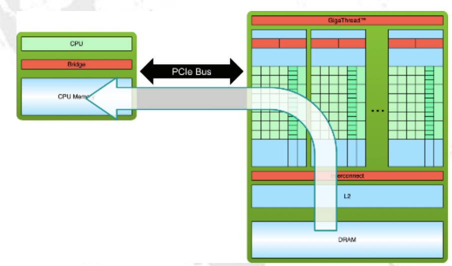

Step 1: CPU allocates and copies data to GPU




Step 2: CPU sends function codes to GPU




Step 3: GPU executes instructions on GPU

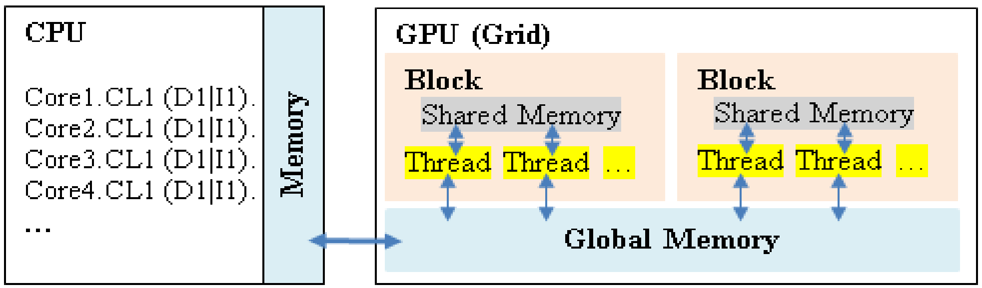



Step 4: Results are copied back to GPU

Asaduzzaman, A., Gummadi, D., and Yip, C.M., “A Talented CPU-to-GPU Memory Mapping Technique,” in IEEE SoutheastCon 2014, Lexington, Kentucky, March 13-16, 2014.

# High Performance Computing:
# A Poisson Solver

## Laplace Equation: Simulation Results

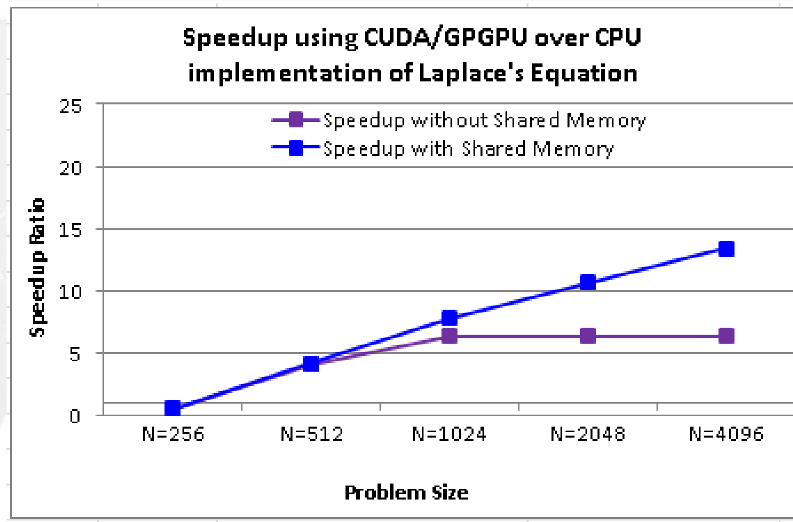



CPU-GPU Memory Organization

| Size N x N x N | CPU Time (sec) | GPGPU Time (sec) | |
|---|---|---|---|
| | | No shared memory | With shared memory |
| N=256 | 1.58 | 3.08 | 2.94 |
| N=512 | 15.57 | 3.84 | 3.77 |
| N=1024 | 130.57 | 20.46 | 16.84 |
| N=2048 | 1783.11 | 279.40 | 167.27 |
| N=4096 | 17206.92 | 2696.20 | 1284.24 |

| CPU | GPGPU |
|---|---|
| • Processor: Intel Xeon E5506<br>• Cores: 2 x Quad-Core<br>• Threads: 2 x 4<br>• Clock Speed: 2.13 GHz<br>• RAM: 8GB DDR3<br>• Max. Memory Bandwidth: 19.2 GB/sec<br>• Power: 80 Watt<br>• OS: Linux (Debian) | • Type: NVIDIA Tesla C2075<br>• Cores: 14 x 32 Cores<br>• RAM: 6GB GDDR5<br>• RAM Speed: 1.5 GHz<br>• RAM Bandwidth: 144 GB/sec<br>• Power: 255 Watt<br>• OS: Not applicable |

CPU-GPU System Parameters

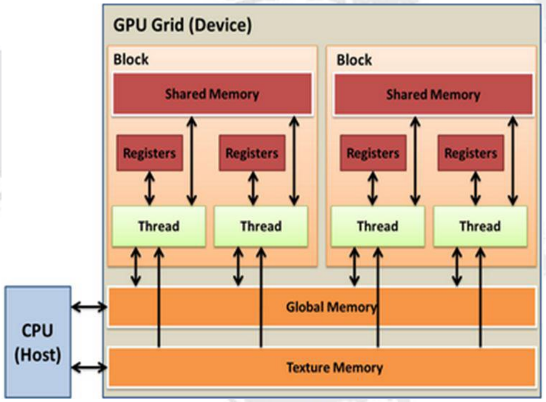



Speedup Vs Data Size

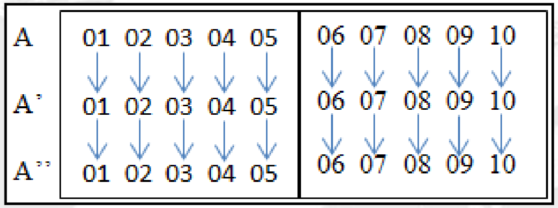
Asaduzzaman, A., Yip, C.M.*, Kumar, S., and Asmatulu, R.; "Fast, Effective, and Adaptable Computer Modeling and Simulation of Lightning Strike Protection on Composite Materials," in IEEE SoutheastCon Conference 2013, Jacksonville, Florida, April 4-7, 2013.

# High Performance Computing:
## “Regrouping Data/Threads for Improving CPU-GPU Performance”

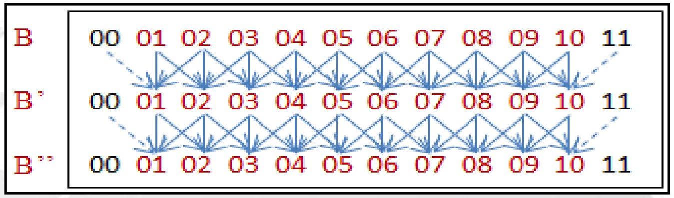
## Regrouping Data



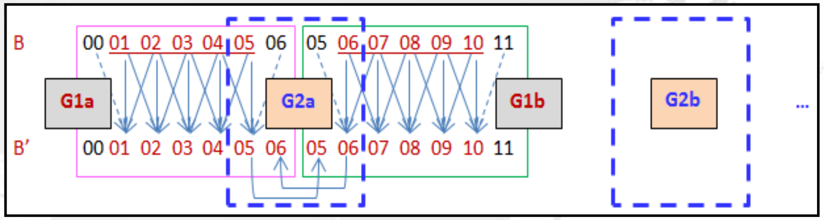
CPU-GPU cache memory subsystem



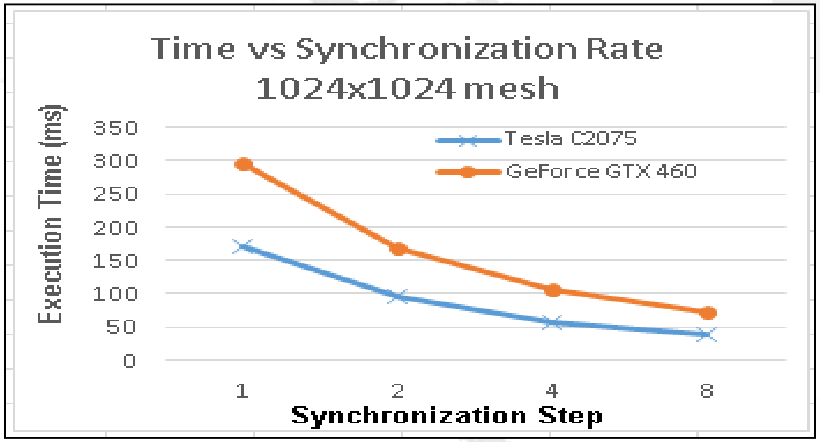
Data without dependency



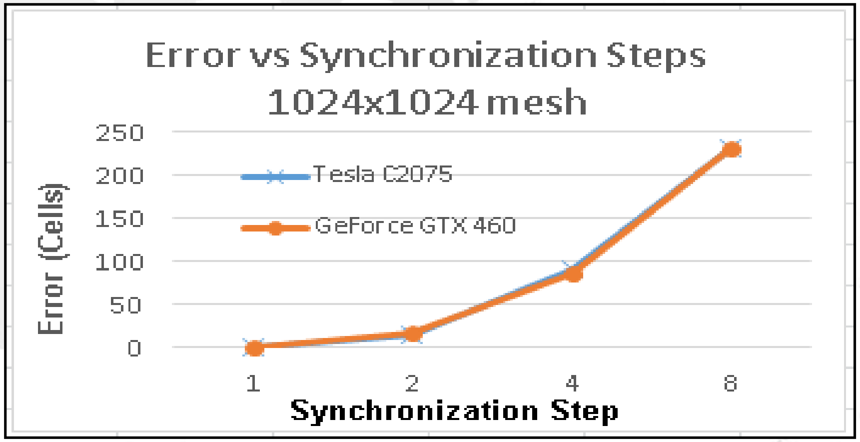
Data with dependency



Processing data with dependency



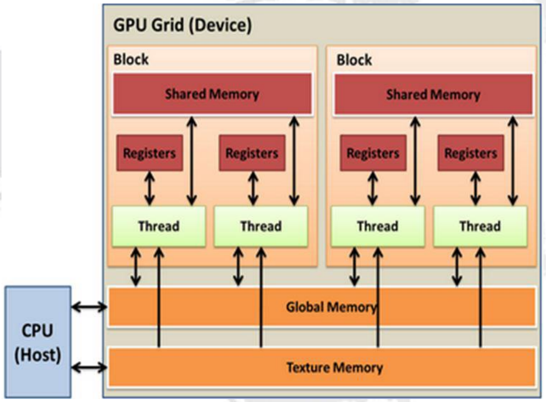
Execution Time Vs Synchronization



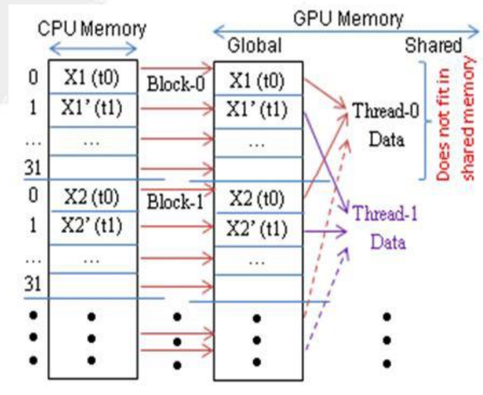
Error (# of cells) Vs Synchronization

# High Performance Computing:
## "Regrouping Data/Threads for Improving CPU-GPU Performance"

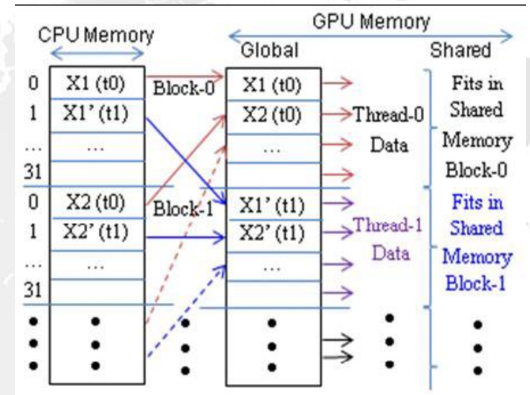
## CPU-to-GPU Memory Mapping



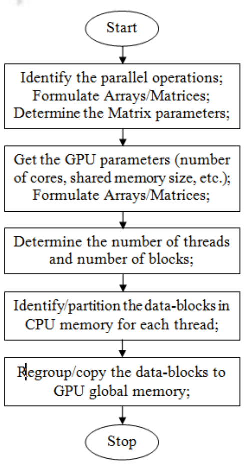

CPU-GPU cache memory subsystem




Typical CPU to GPU memory map




Proposed CPU to GPU memory map




Major Steps

| Parameter | Description |
|---|---|
| CPU | Intel Xeon |
| CPU Cores | 8 |
| CPU RAM | 6GB |
| Fermi GPU Card | NVIDIA Tesla C2075 |
| Fermi GPU Cores | 448 |
| Fermi Clock Speed | 1.15 GHz |
| Fermi Global Memory | 5.4GB |
| Fermi Shared Memory | 49KB/Block |
| Kepler GPU Card | NVIDIA Tesla K20m |
| Kepler GPU Cores | 2496 |
| Kepler Clock Speed | 0.71 GHz |
| Kepler Global Memory | 4.8GB |
| Kepler Shared Memory | 49KB/Block |
| Operating System | Linux Debian |

System Parameters

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 10000 | 10000 | 0 | 0 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 10000 | 10000 | 0 | 0 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

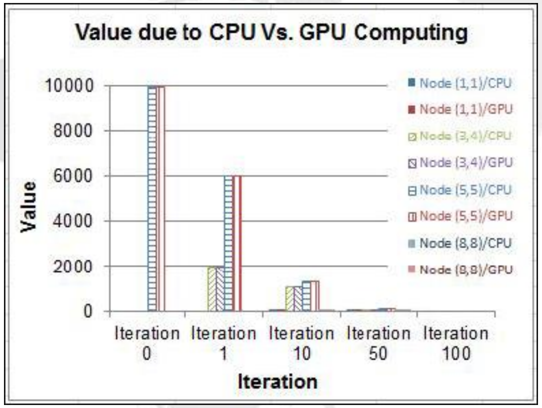
| 4 | 5 | 6 | 7 | 8 | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 10000 | 10000 | 0 | 0 | 0 | 0 | 4 |
| 10000 | 10000 | 0 | 0 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | |

Validation: C vs CUDA/C Results

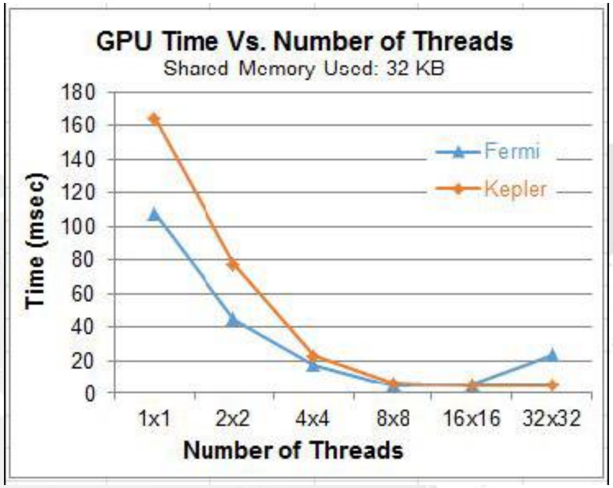
Asaduzzaman, A., Gummadi, D., and Yip, C.M., "A Talented CPU-to-GPU Memory Mapping Technique," in IEEE SoutheastCon 2014, Lexington, Kentucky, March 13-16, 2014.

# High Performance Computing:
## "Regrouping Data/Threads for Improving CPU-GPU Performance"

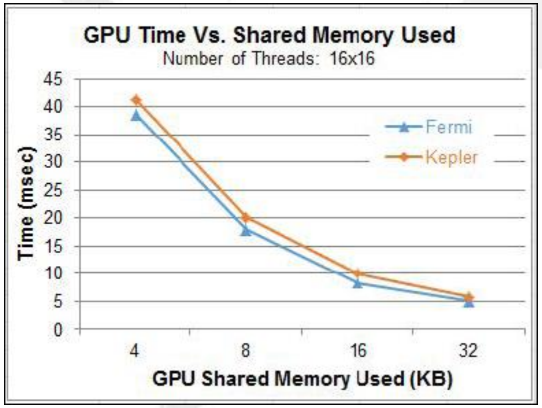
## Simulation Results



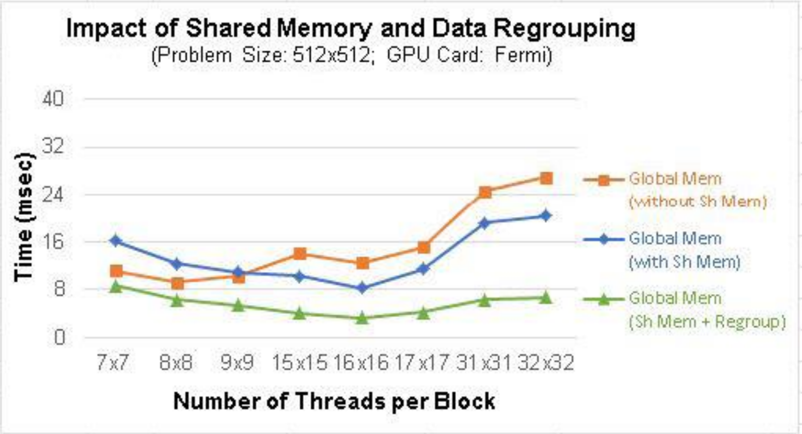

Validation: C Vs CUDA/C Results




GPU Time Vs # of Threads




GPU Time Vs Shared Memory




GPU Time Vs # of Threads per Block

Asaduzzaman, A., Gummadi, D., and Yip, C.M., "A Talented CPU-to-GPU Memory Mapping Technique," in IEEE SoutheastCon 2014, Lexington, Kentucky, March 13-16, 2014.

# High Performance Computing:
## "A Communication-Aware Cache-Controller for HPC Systems"

To offer low-cost low-power high performance computing (HPC), multicore central processing unit (CPU)/many-core graphics processing unit (GPU) architectures have become the mainstream processor design choice.

No operating system on GPU cards → no flexibility to write parallel programs.

… having hundreds or thousands of cores on a CPU …

- high core-to-core communication delay
- High synchronization delay
- poor scalability

Other problems include

- high power consumption → heat

Root causes of the problems include

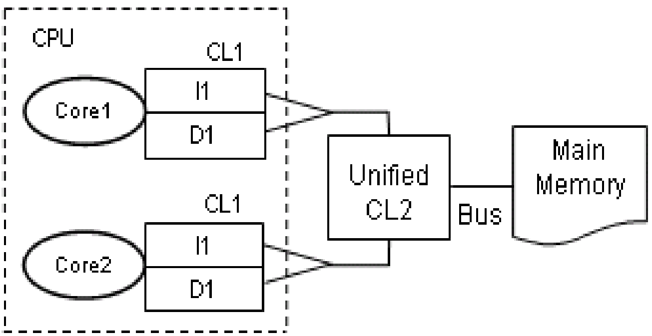
- The traditional power-hungry dynamic characteristics of multi-level caches
- The wired network topologies

This project aims to enhance the scalability of multicore/many-core systems by maneuvering the cache subsystem and normalizing the parallelism in multithreading. The proposed level-2 cache-mediator (L2CM) assists in minimizing memory latency and synchronization delay.

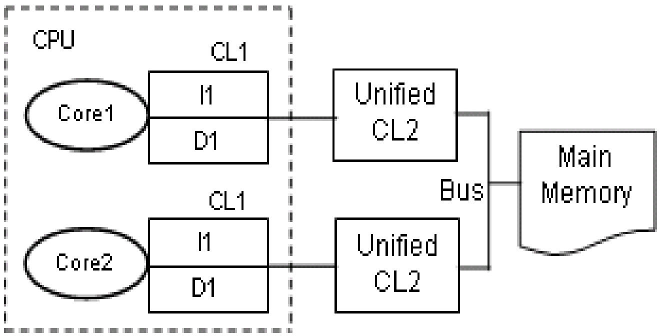
Asaduzzaman, A. and Sibai, F.N.*, "A Communication-Aware Cache-Controller to Minimize the HPC Communication Latency," under preparation, IEEE Transactions on Parallel and Distributed Systems (TPDS), 2020.

# High Performance Computing:
## "A Communication-Aware Cache-Controller for HPC Systems"

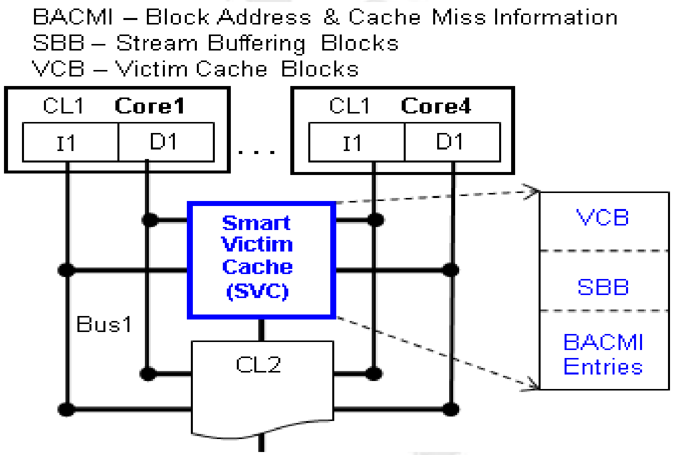
### A novel multicore architecture with victim cache block



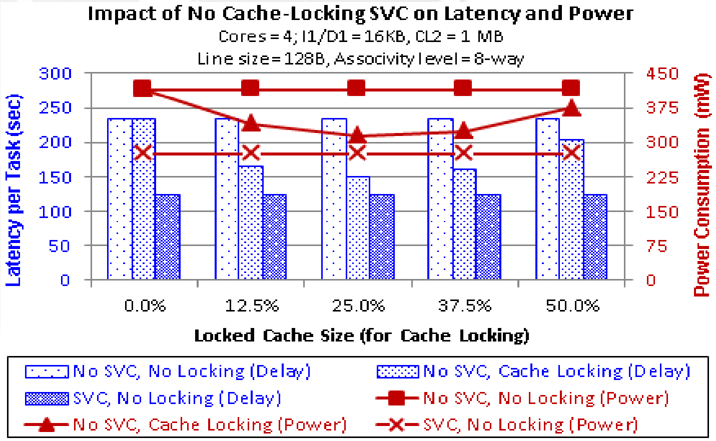

Memory Subsystem with Shared CL2




Memory Subsystem with Dedicated CL2



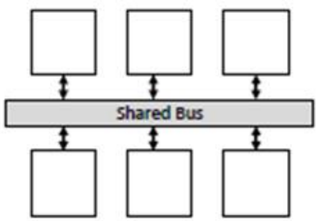

Proposed architecture with SVC

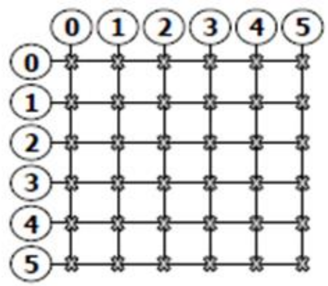



Improvement in latency and power consumption

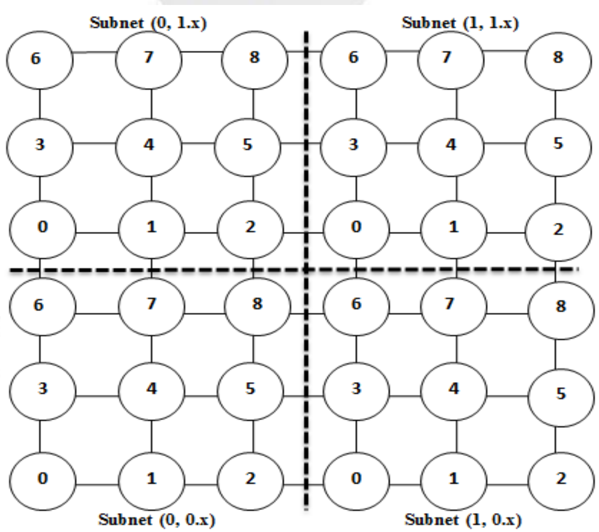
Jareen, T., "Improving Performance, Power, and Security of Multicore Systems Using Smart Cache Organization," MS Thesis, Wichita State University, 2014. (Advisor: Asaduzzaman, A.)

## DASH-Like Multicore WNoC Arch. to Minimize Latency.

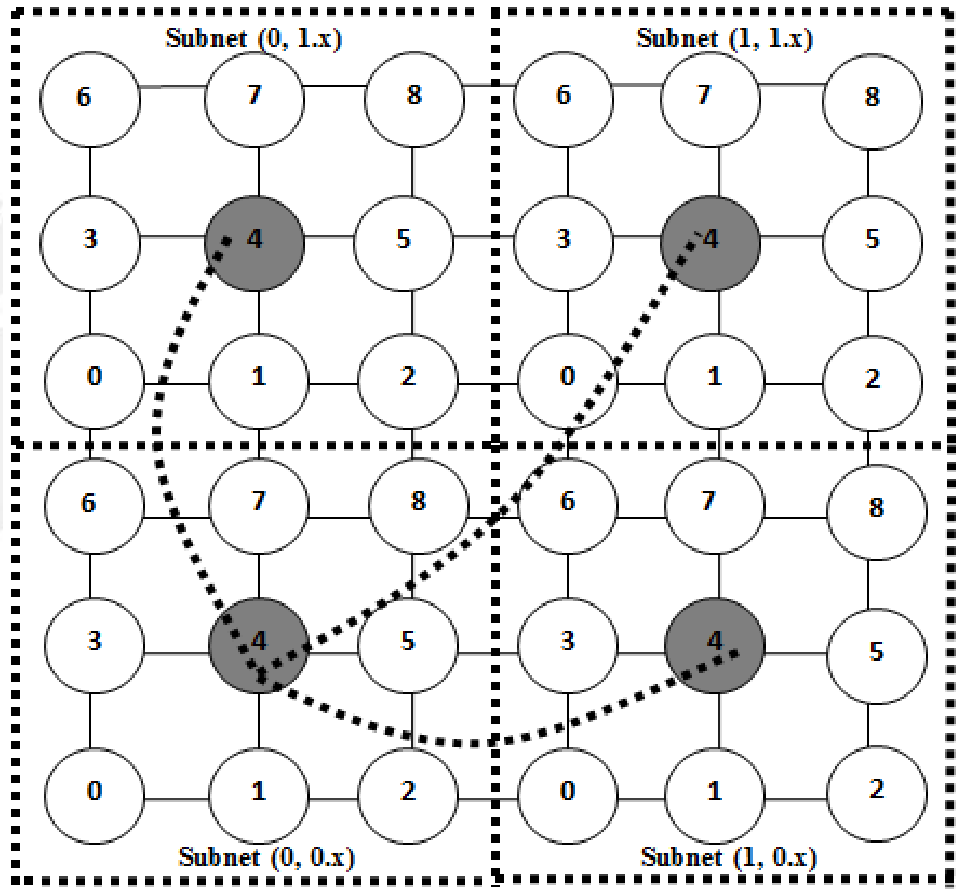



Bus and Crossbar Network Topologies



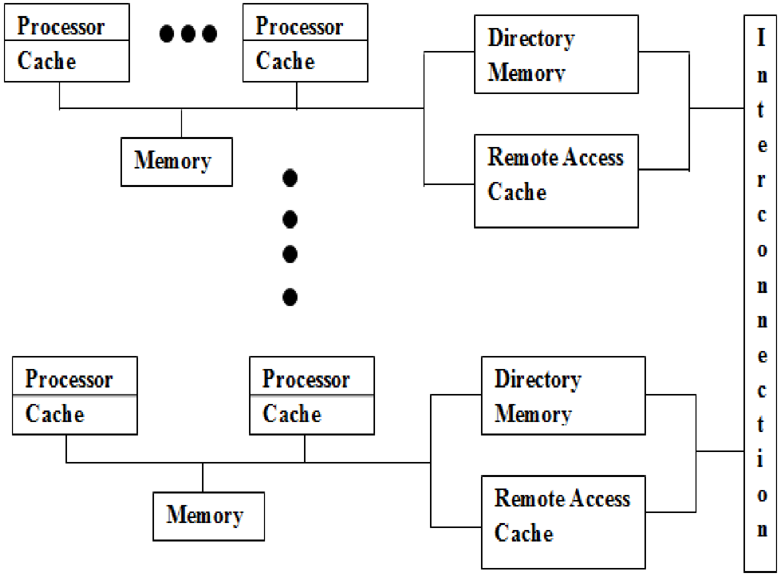

Subnet in a Mesh



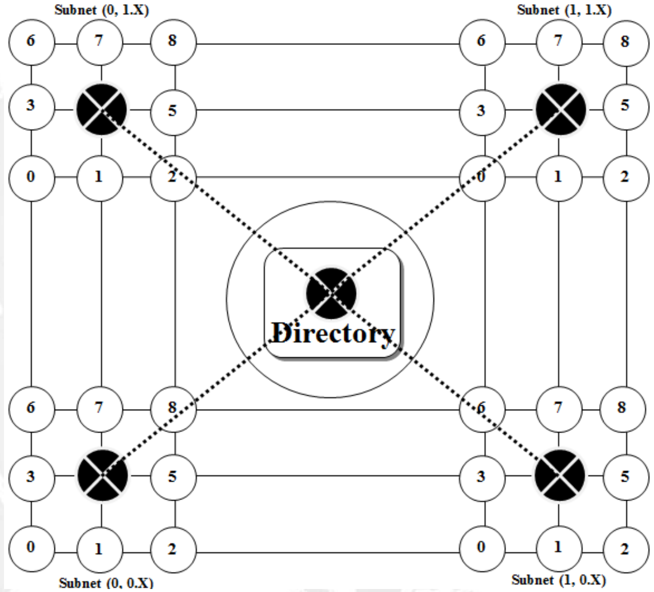

Wireless network on-chip (WNoC) architecture


Asaduzzaman, A., Chidella, K.K.*, and Vardha, D.*, “An Energy-Efficient Directory Based Multicore Architecture with Wireless Routers to Minimize the Communication Latency,” IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 28, No. 2, pp. 374-385, May 2016.

# DASH-Like Multicore WNoC Arch. to Minimize Latency.




A Directory Architecture for SharedMemory (DASH) Multiprocessor System




Proposed architecture with a centralized directory and wireless routers

A ROW IN THE DIRECTORY REPRESENTS THE INITIAL STAGE OF CORE-1

| Core # | Block 0 | Block 1 | Block 2 | Block 3 | Block 4 | Block 5 | Block 6 | Block 7 |
|---|---|---|---|---|---|---|---|---|
| Core1 (0, 0.0) | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty | 0 Addr Empty |

Designing directory of the proposed WNoC architecture using a MESI-like protocol

Asaduzzaman, A., Chidella, K.K.*, and Vardha, D.*, "An Energy-Efficient Directory Based Multicore Architecture with Wireless Routers to Minimize the Communication Latency," IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 28, No. 2, pp. 374-385, May 2016.

# High Performance Computing:
## "A Communication-Aware Cache-Controller for HPC Systems"

## DASH-Like Multicore WNoC Arch. to Minimize Latency.

Simulation:

SOURCE AND DESTINATION NODES FOR DIFFERENT COMMUNICATION CASES

| Case Number | Source Node (S) | Destination Node (D) | Remark |
|---|---|---|---|
| Case 1 | Core - > (0, 0.0) | Core - > (1, 1.8) | S and D are in different subnets |
| Case 2 | Core - > (0, 0.4) | Core - > (1, 1.4) | S and D are in different subnets |
| Case 3 | Core - > (0, 0.7) | Core - > (1, 0.1) | S and D are in different subnets |
| Case 4 | Core - > (0, 0.3) | Core - > (0, 1.5) | S and D are in different subnets |
| Case 5 | Core - > (1, 0.5) | Core - > (0, 1.2) | S and D are in different subnets |

COMMUNICATION LATENCY FOR THREE DIFFERENT ARCHITECTURES

| Different Scenarios | Mesh (multicasting) | WNoC Architecture | Proposed Architecture |
|---|---|---|---|
| Case 1: (0,0.0)-(1,1.8) | 4x9+40=76 | 4x4+40=56 | 4x2+40=48 |
| Case 2: (0,0.4)-(1,1.4) | 4x5+40=60 | 4x0+40=40 | 4x0+40=40 |
| Case 3: (0,0.7)-(1,0.1) | 4x4+40=56 | 4x2+40=48 | 4x1+40=44 |
| Case 4: (0,0.3)-(0,1.5) | 4x4+40=56 | 4x2+40=48 | 4x1+40=44 |
| Case 5: (1,0.5)-(0,1.2) | 4x4+40=56 | 4x3+40=52 | 4x1+40=44 |

TOTAL POWER CONSUMPTION FOR THREE ARCHITECTURES

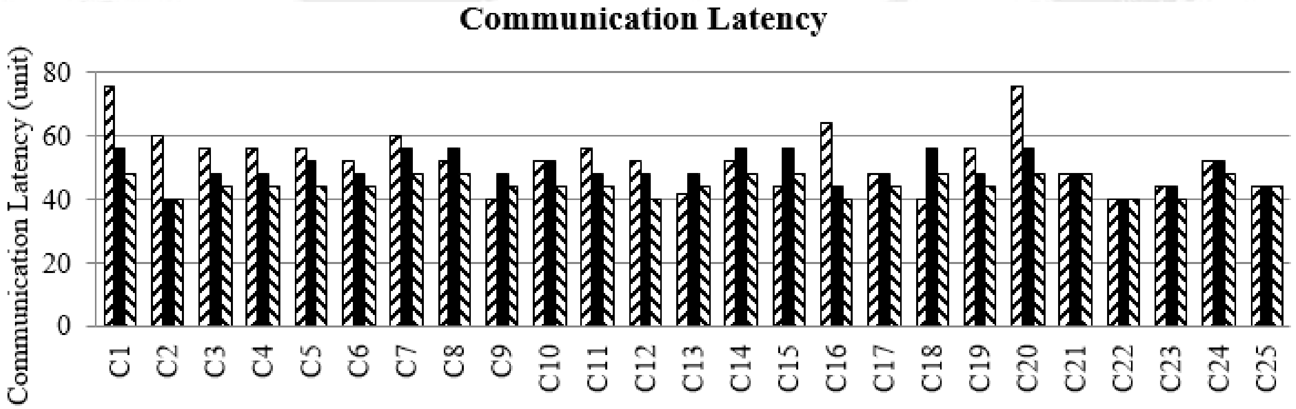
| Different Scenarios | Mesh Architecture | WNoC Architecture | Proposed Architecture |
|---|---|---|---|
| Case 1: (0,0.0)-(1,1.8) | Ptot = P1 + P2+ P3<br>P1 = (Pwr*Nwr) +<br>(Pcwr*Ncwr) = 43<br>P2 = (Pwr*Nwr) +<br>(Pcwr*Ncwr) = 43<br>P3 = Palwr+ Pcanw<br>= 5.5 + 19.5 = 25 | Ptot = Psd + Pds<br>Psd=Pawrsn+(Pcwr*Ncwr)<br>+Pcwl+3(Pwl+Pcasn)<br>=2.5+(3*2)+3.3+25.8 = 37.6<br>Pds = Pdsn+Pwl+Pssn<br>= 11.8+1.1+11.8<br>= 24.7 | Ptot = Psdr + Pcdr<br>Psdr=Pawrsn+(Pcwr*Nc<br>wr)+Pcwl+Pwl<br>=2.5+(3*2)+3.3+1.1<br>= 12.9<br>Pcdr = Pdr+Pcwl<br>= 6 + 3.3 = 9.3 |

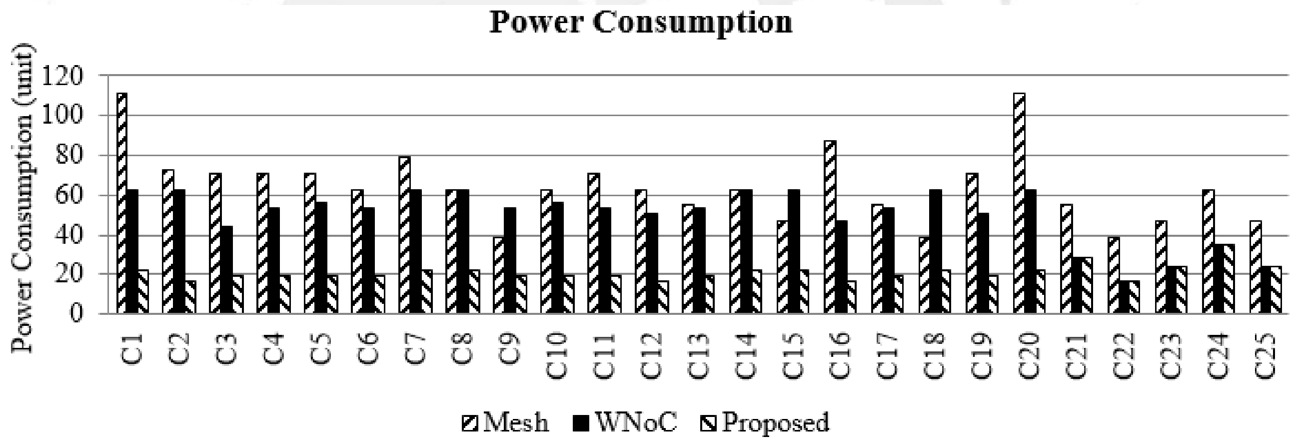
Asaduzzaman, A.,
the Communicatio

# High Performance Computing:
## "A Communication-Aware Cache-Controller for HPC Systems"

## DASH-Like Multicore WNoC Arch. to Minimize Latency.

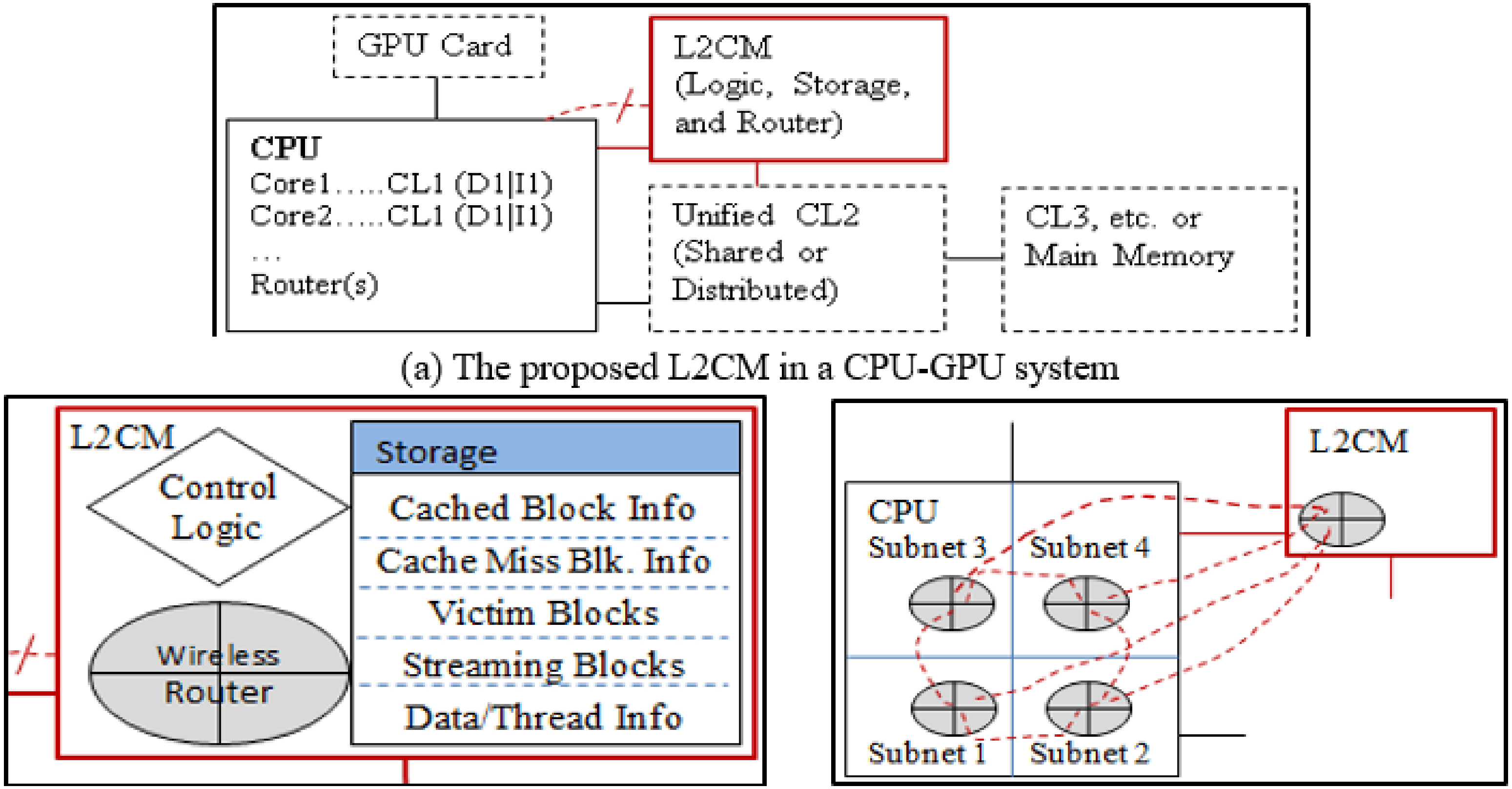
Results:




Asaduzzaman, A., Chidella, K.K.*, and Vardha, D.*, "An Energy-Efficient Directory Based Multicore Architecture with Wireless Routers to Minimize the Communication Latency," IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 28, No. 2, pp. 374-385, May 2016.

# A Level-2 Cache-Mediator for Enhancing Scalability




(a) The proposed L2CM in a CPU-GPU system

(b) The proposed L2CM subsystem

(c) Communication module for the system

Architectural layout of a CPU-GPU system with the proposed L2CM

Asaduzzaman, A. and Sibai, F.N.*, “A Communication-Aware Cache-Controller to Minimize the HPC Communication Latency,” under preparation, IEEE Transactions on Parallel and Distributed Systems (TPDS), 2020.

# A Level-2 Cache-Mediator for Enhancing Scalability

**L2CM control logic** can be used to assist the CPU with the following tasks:

*Searching for any block x:*
In case of a CL1 miss, other CL1s should be checked first to find the required block x.
    If x is in {Cached Block Info}, then
        Satisfy the request from one of the CL1s;
    Else check L2CM and CL2 at the same time
        If found in L2CM, swap CL1 block with L2CM block; done;
        If found in CL2, normal cache activities…
        Else issue stream buffering call (say, from main memory).

*Selecting CL1 victim block x:*
If CL1 is full and needs to bring in a new block, then a victim block is selected to replace the new block.
    x ← using Cache Miss Block Information, find the one with the minimum cache misses.

*Determining if CL1 victim block x should be stored in L2CM as a Victim Block:*
    If x is NOT in {L2CM Victim Blocks}, then
        'b' ← the L2CM Victim Block that has the minimum cache misses
        If (number of cache misses in x) > (number of cache misses in b), then
            L2CM victim block x should be stored in L2CM; replace b with x.

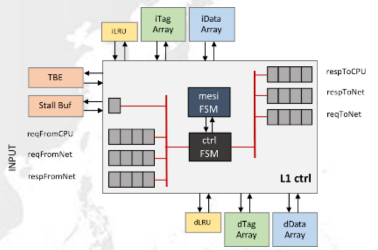
*Regrouping data/threads for GPU computing:*
    Step 1: Find the number of cores on the GPU card.
    Step 2: Identify the parallel segments (let's call them subproblems) of the problem.
    Step 3: Determine the data-size of the subproblem (such as number of rows and columns in a mesh).
    Step 4: Determine the number of computations and the optimal number of threads.

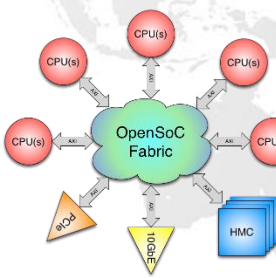
# High Performance Computing:
## "A Communication-Aware Cache-Controller for HPC Systems"

"Open2C framework and OpenSoC Fabric to build up a communication-aware level-2 cache controller"

2020 SUMMER RESEARCH AT BERKELEY LAB2019 Sustainable Research Pathways Workshop → Matched with Berkeley Lab Researcher

This project aims to use the Berkeley Lab Open Cache Coherence (Open2C) framework and OpenSoC Fabric to build up a communication-aware level-2 cache controller (CAL2CC). This project is expected to provide a good way for exercising the Open2C and OpenSoC. The proposed CAL2CC has potential to enhance the scalability of multicore systems by maneuvering their cache memory subsystems.




Asaduzzaman, A., "Open2C framework and OpenSoC Fabric to build up a communication-aware level-2 cache controller," proposal submitted for 2020 SUMMER RESEARCH AT BERKELEY LAB, 2019 Sustainable Research Pathways Workshop, matched with Dr. John Shalf
https://crd.lbl.gov/departments/computer-science/cag/research/open2c/
https://crd.lbl.gov/departments/computer-science/cag/research/opensoc-fabric/

# High Performance Computing ?

- HPC or Supercomputing?

| *Consideration* | *HPC* | *Supercomputing* | *Note* |
|---|---|---|---|
| Processing Cores | ~5,000 (CPU, GPU, homogeneous) | ~300,000 (CPU, GPU, heterogeneous) | |
| Processing Power | Tera (10^12) FLOPS | Peta (10^15) FLOPS | |
| Power (Energy) | Low (~430 W) | High (~4.04 MW) | |
| Price | Low (~$5K) | High (10x M$) | |
| Analogy: Cars | Formula One Race cars | Special, Expensive Race cars | |

# "High Performance Computing, Machine Learning, and Big Data Analytics for Common Good"

## Outline

QUESTIONS? Any time, please!

- **Introduction**
  - About Myself (Asaduzzaman)
  - Computational Systems: Past, Present, and Future
- **High Performance Computing (HPC)**
  - Hybrid HPC Systems and Parallel Computing/Programming
  - "Regrouping Data/Threads for Improving CPU-GPU Performance"
  - "A Communication-Aware Cache-Controller for HPC Systems"
- **Machine Learning (ML): Medical Image Processing**
  - "Real-Time Image Processing for Breast Cancer Treatment"
- **Geospatial Big Data (BD) Analytics using HPC and ML**
  - "Geospatial Cyberinfrastructure for Common Good"
- **Q/A: Discussion**

# Machine Learning (ML): Medical Image Processing

## "Real-Time Image Processing for Breast Cancer Treatment"

Poor contrasts of mammogram images may lead mistakes while treating breast cancer patients.

Introduce a novel imaging technique … using the numerical pixel-values and the hidden attributes of the target mammogram images.

The extracted feature values are split into training and testing sets. ML

Pattern recognition techniques, namely the Linear Discriminant Analysis (LDA) method and Support Vector Machine (SVM) model, are used for training and testing purposes.

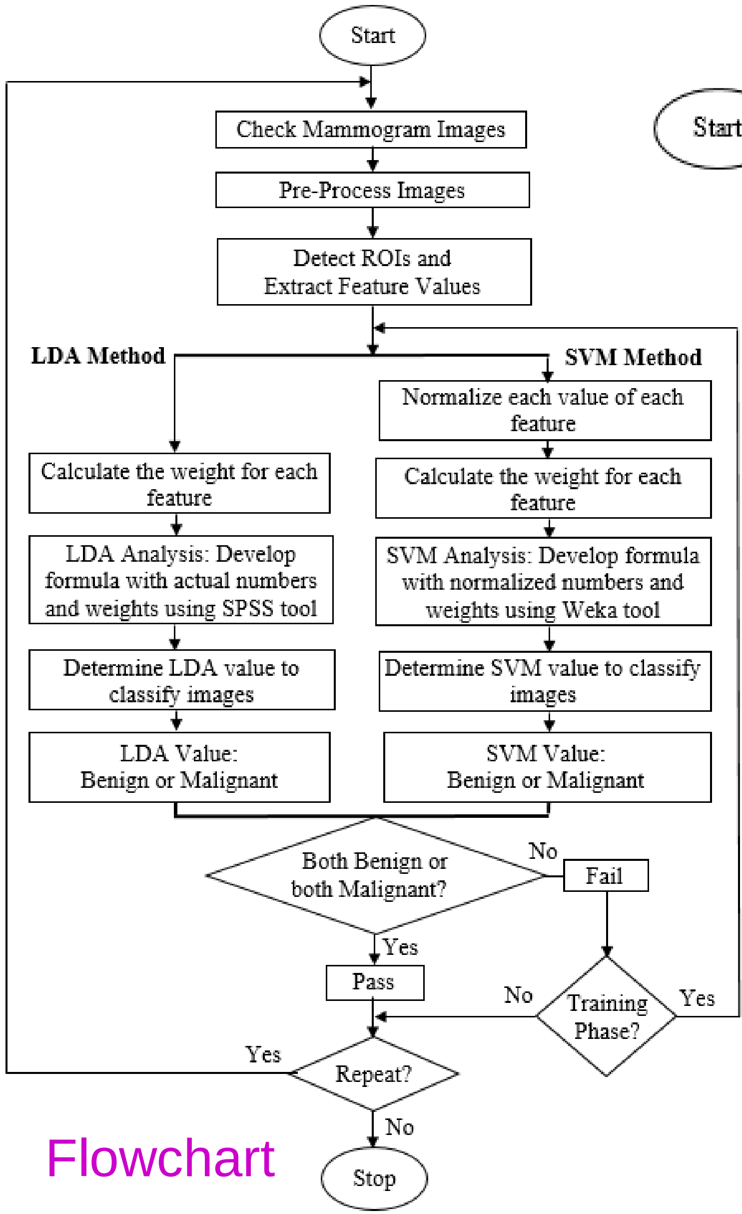
Images from the Digital Database for Screening Mammogram (DDSM) and Mammographic Image Analysis Society (MIAS) are used.

Experimental results using 1505 (1000 + 505) images, we observe a 100% taxonomy rate of identifying benign and malignant cells of the mammogram images.

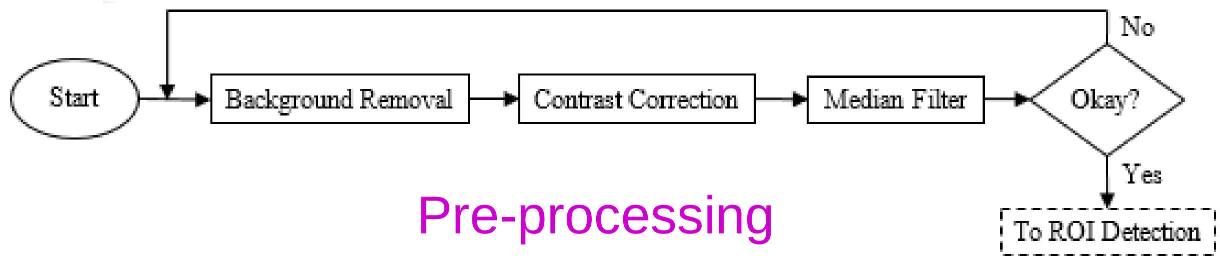
Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., "An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis," under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

# Machine Learning (ML): Medical Image Processing

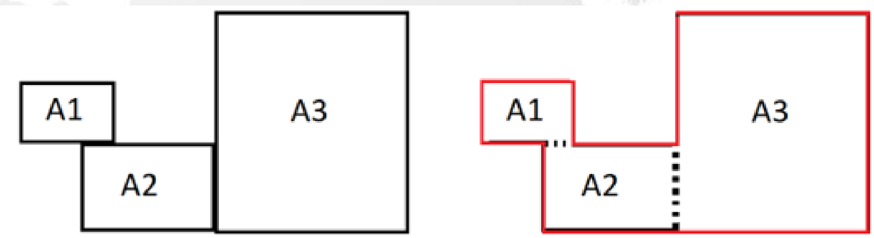
## “Real-Time Image Processing for Breast Cancer Treatment”




Flowchart




Pre-processing



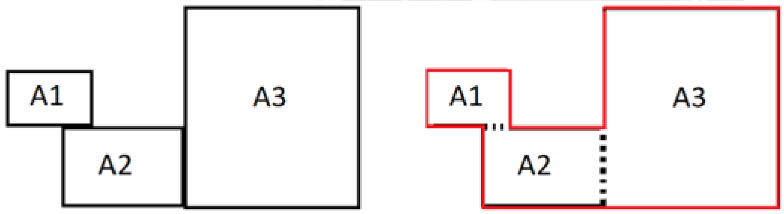

(a) ROI Area = Area A1 + Area A2 + Area A3 (b) Perimeter is indicated by the solid line

Region of Interest (ROI) Detection

K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast
:w, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No.

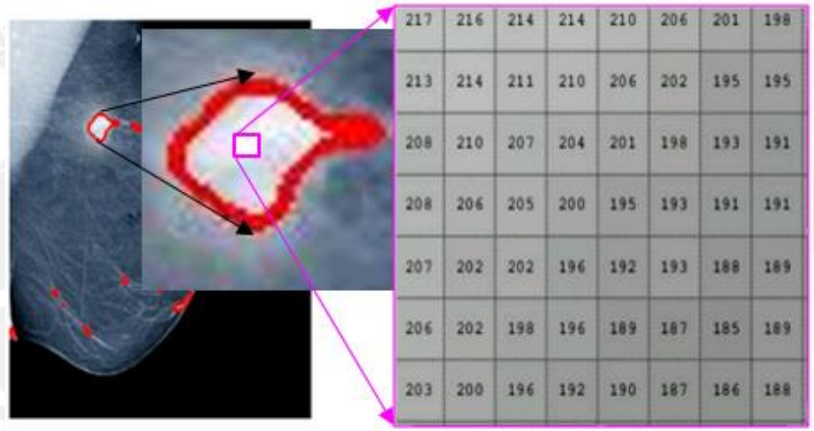
# Machine Learning (ML): Medical Image Processing

## “Real-Time Image Processing for Breast Cancer Treatment”




(a) ROI Area = Area A1 + Area A2 + Area A3 (b) Perimeter is indicated by the solid line

Region of Interest (ROI) Detection



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 217 | 216 | 214 | 214 | 210 | 206 | 201 | 198 |
| 213 | 214 | 211 | 210 | 206 | 202 | 195 | 195 |
| 208 | 210 | 207 | 204 | 201 | 198 | 193 | 191 |
| 208 | 206 | 205 | 200 | 195 | 193 | 191 | 191 |
| 207 | 202 | 202 | 196 | 192 | 193 | 188 | 189 |
| 206 | 202 | 198 | 196 | 189 | 187 | 185 | 189 |
| 203 | 200 | 196 | 192 | 190 | 187 | 186 | 188 |

A 7x8 matrix of a ROI with 7x8 pixels

### Feature Extraction

**Geometrical Features**

Area, Perimeter, and Radius

$$Area\ (of\ a\ circular\ shape) = \pi r^2 \ldots (1a)$$

$$Area\ (of\ an\ irregular\ shape) = \textstyle\sum_i \sum_j A_{i,j} \ldots (1b)$$

$$Perimeter\ (of\ a\ circular\ shape) = 2\pi r \ldots (2a)$$

$$Perimeter\ (of\ an\ irregular\ shape) = \textstyle\sum_i \sum_j P_{i,j} \ldots (2b)$$

**Textural Features**

Mean Value, Global Mean, and Standard Deviation, Entropy, and Skewness

$$\text{Entropy} = -\textstyle\sum (\text{I} \times \log_2(\text{I})) \ldots (3)$$

Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis,” under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

# Machine Learning (ML): Medical Image Processing

## “Real-Time Image Processing for Breast Cancer Treatment”

### Data Analysis

LDA Value = Value1 x Weight1 + Value2 x Weight2 + ...

+ Final-Value x Final-Weight ..................... (4)

SVM Value = Normalized-Value1 x Weight1 + Normalized-Value2 x Weight2 + ...

+ Normalized-Final-Value x Final-Weight ........................... (5)

### Mammogram Images Used

From DDMS 2620 and MIAS 322 images, we used 1383 DDMS and MIAS 122 images.

Sample of Mammogram Images Considered

| *Image* | *Source* | *Category* | *Remark* |
|---|---|---|---|
| Mdb021.pgm | MIAS | Benign | False positive at bottom |
| Mdb032.pgm | MIAS | Benign | Normal breast |
| Mdb063.pgm | MIAS | Benign | Benign mass at middle |
| Mdb091.pgm | MIAS | Benign | Dense normal breast |
| Mdb148.pgm | MIAS | Malignant | Speculated masses |
| Mdb179.pgm | MIAS | Malignant | Tumor on entire breast |
| Mdb184.pgm | MIAS | Malignant | Tumor is clearly visible |
| Mdb202.pgm | MIAS | Malignant | Big tumor at middle |
| D-1077-1 | DDMS | Malignant | False positive tumor |
| D-4032-1 | DDMS | Malignant | Malignant mass |
| D-4126-1 | DDMS | Malignant | Tumor – first stage |
| D-4141-1 | DDMS | Malignant | Malignant tumor |

### Tools Languages Used:

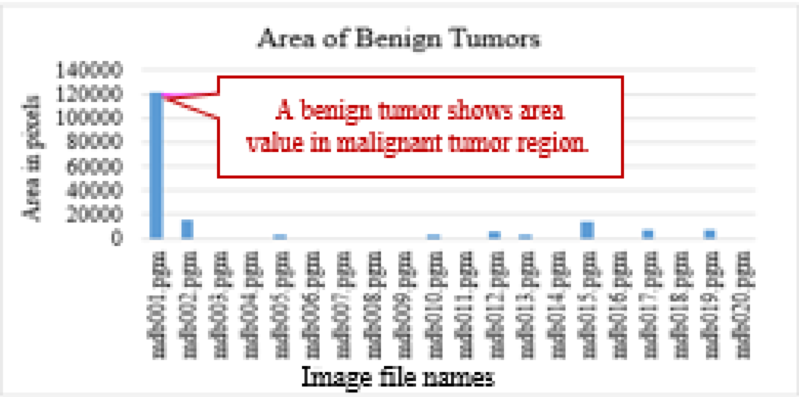
Photomania DX, MATLAB, LDA, SVM, Excel

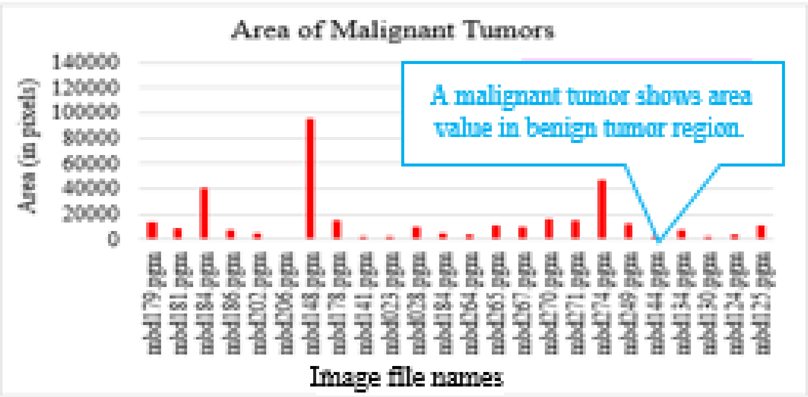
Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis,” under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

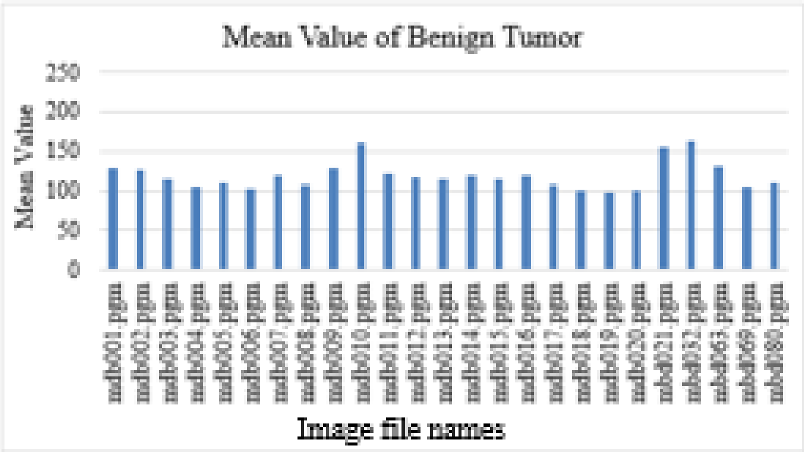
# Machine Learning (ML): Medical Image Processing

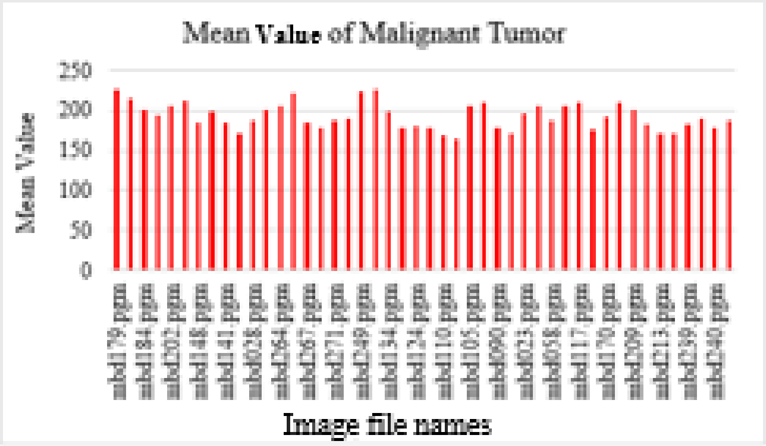
## “Real-Time Image Processing for Breast Cancer Treatment”

## Experimental Results



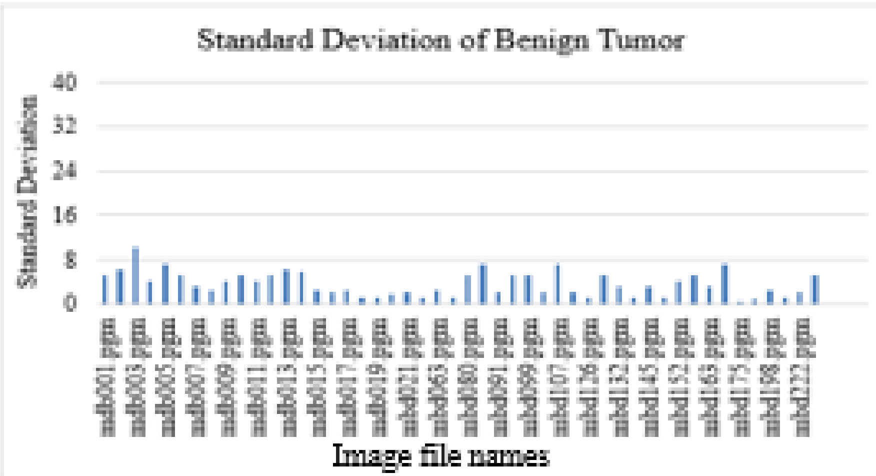



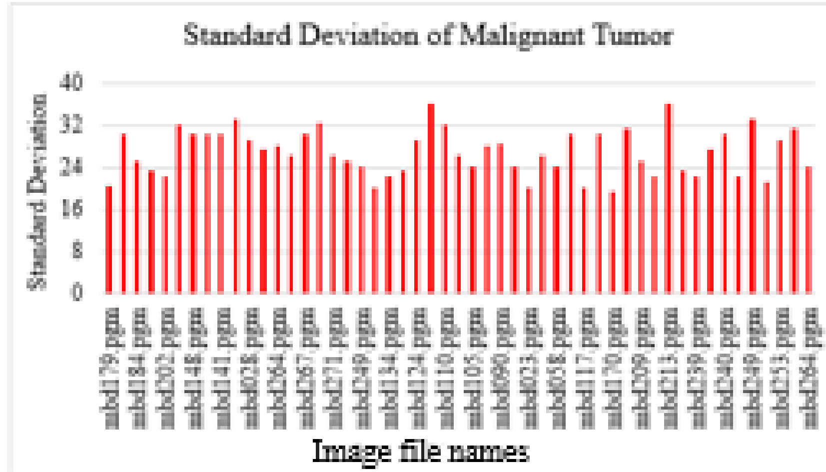



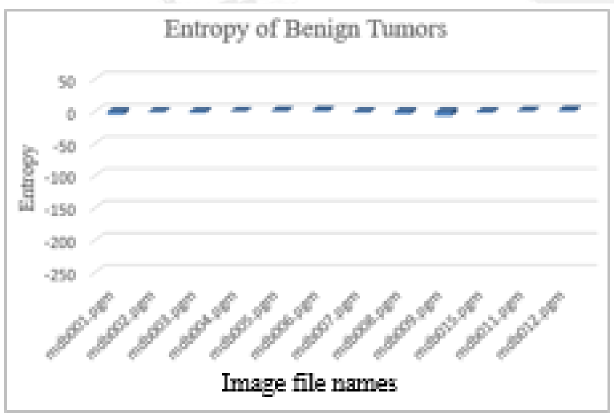

Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis,” under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

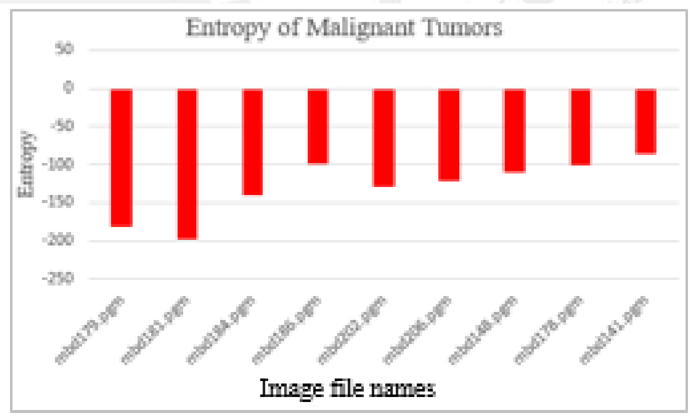
# Machine Learning (ML): Medical Image Processing
## "Real-Time Image Processing for Breast Cancer Treatment"

## Experimental Results




Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., "An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis," under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

# Machine Learning (ML): Medical Image Processing
## "Real-Time Image Processing for Breast Cancer Treatment"

## Experimental Results

LDA Value (MIAS) = 0.026 x Mean-Value + 0.179 x Standard-Deviation –
0.019 x Entropy – 0.017 x Skewness – 7.887 ................................ (6)

LDA Value (DDSM) = 0.022 x Mean-Value – 0.226 x Global-Mean +
0.069 x Standard-Deviation – 0.024 x Entropy –
0.021 x Skewness – 6.076 ................................................ (7)

SVM Value (MIAS) = –0.2575 x Normalized-Area + 1.3209 x Norm-Perimeter –
0.1538 x Norm-Radius + 0.9799 x Norm-Mean-Value +
0.1819 x Norm-Global-Mean + 1.6045 x Norm-Std-Deviation –
1.9652 x Norm-Entropy – 0.4783 x Norm-Skewness + 0.3456 ......... (8)

SVM Value (DDSM) = 1.3406 x Normalized-Mean-Value –
0.4456 x Norm-Global-Mean + 1.7355 x Norm-Std-Deviation –
2.6728) x Norm-Entropy – 0.1070 x Norm-Skewness + 0.5316 ......... (9)

Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., "An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis," under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

# Machine Learning (ML): Medical Image Processing

## “Real-Time Image Processing for Breast Cancer Treatment”

## Experimental Results

Illustration of LDA and SVM methods to separate MIAS images into benign or malignant

| Known Decision | Classification of MIAS Images using Texture Values | | | | | | | | | | LDA Value | SVM Value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean Value | | Global Mean | | Std. Deviation | | Entropy | | Skewness | | | |
| | Actual | Norm. | Actual | Norm. | Actual | Norm. | Actual | Norm. | Actual | Norm. | | |
| Benign | 130.00 | -0.804 | 1.022 | -1.038 | 5.232 | -0.825 | -5.213 | 0.863 | -1.235 | -0.804 | -3.450 | -3.638 |
| Benign | 126.00 | -0.912 | 1.034 | -1.029 | 6.236 | -0.742 | -1.236 | 0.920 | 38.256 | 1.608 | -4.122 | -4.371 |
| Benign | 115.00 | -1.208 | 1.016 | -1.043 | 10.235 | -0.410 | -2.327 | 0.904 | 36.236 | 1.484 | -3.637 | -4.424 |
| Benign | 105.00 | -1.478 | 1.050 | -1.015 | 4.266 | -0.905 | 0.254 | 0.941 | 0.124 | -0.721 | -4.400 | -4.495 |
| ... | | | | | | | | | | | | |
| Malignant | 184.00 | 0.651 | 1.041 | 1.105 | 30.213 | 1.247 | -136.33 | -1.021 | -2.362 | -0.873 | 4.935 | 5.759 |
| Malignant | 178.00 | 0.489 | 1.005 | 0.972 | 32.325 | 1.422 | -140.36 | -1.079 | 3.236 | -0.531 | 5.139 | 5.796 |
| Malignant | 188.00 | 0.759 | 1.032 | 1.222 | 26.320 | 0.924 | -97.33 | -0.461 | 4.325 | -0.464 | 3.488 | 4.087 |
| Malignant | 190.00 | 0.813 | 0.973 | 0.959 | 25.236 | 0.834 | -165.33 | -1.438 | 3.250 | -0.530 | 4.656 | 5.773 |
| ... | | | | | | | | | | | | |

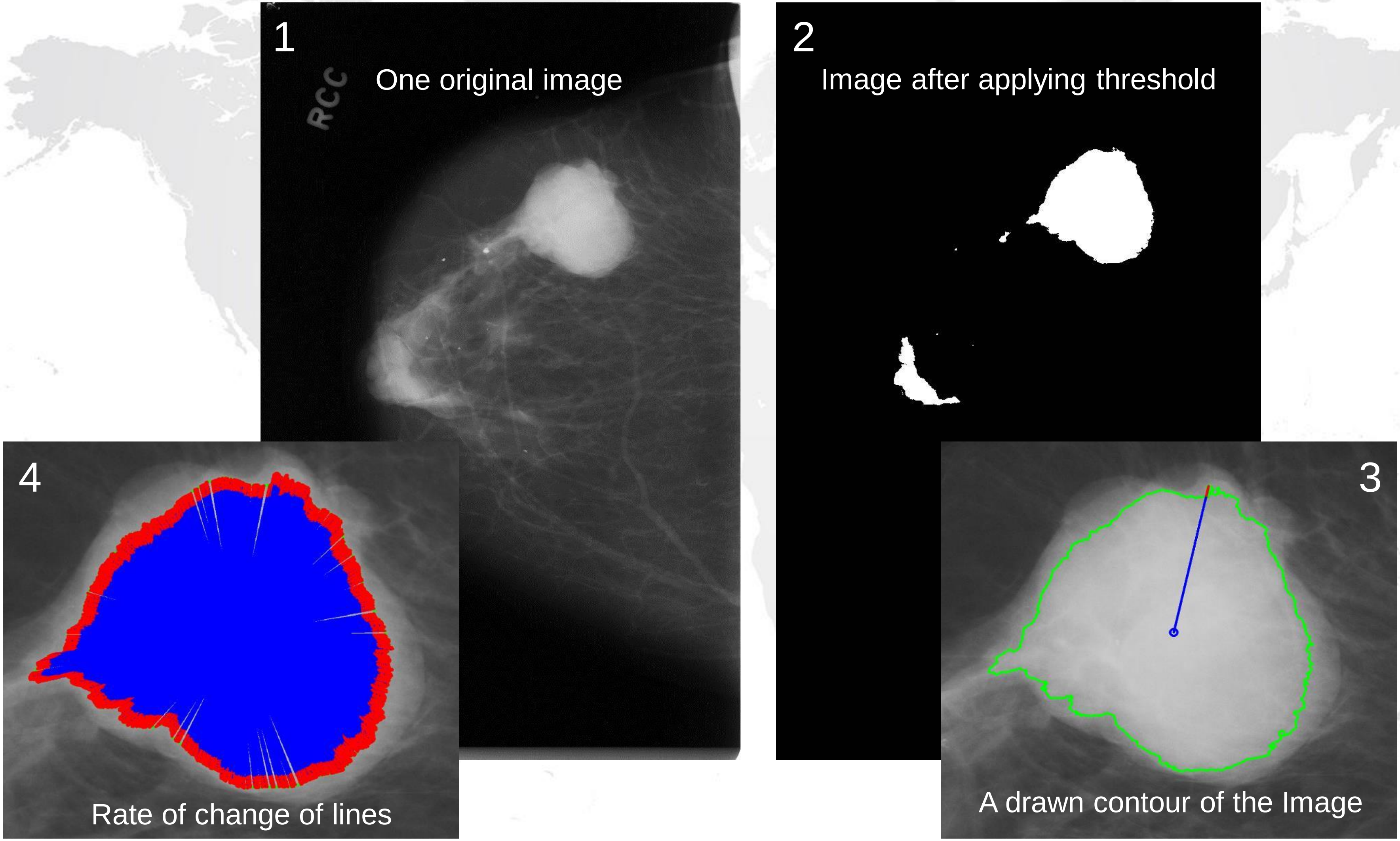
Illustration of LDA and SVM methods to separate DDSM images into benign or malignant

| Known Decision | Classification of DDSM Images using Texture Values | | | | | | | | | | LDA Value | SVM Value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean Value | | Global Mean | | Std. Deviation | | Entropy | | Skewness | | | |
| | Actual | Norm. | Actual | Norm. | Actual | Norm. | Actual | Norm. | Actual | Norm. | | |
| Benign | 153.00 | -0.080 | 0.236 | -1.241 | 2.00 | -1.096 | -6.33 | 1.000 | 40.230 | -0.039 | -3.318 | -3.593 |
| Benign | 89.00 | -1.732 | 0.460 | -1.106 | 3.60 | -0.994 | -16.24 | 0.858 | 19.236 | -0.070 | -3.988 | -5.307 |
| Benign | 162.00 | 0.153 | 0.980 | -0.791 | 1.30 | -1.141 | -10.32 | 0.943 | 10.326 | -0.083 | -2.613 | -3.402 |
| Benign | 113.00 | -1.113 | 1.190 | 0.664 | 2.40 | -1.071 | -19.24 | 0.814 | 20.370 | -0.068 | -3.660 | -4.691 |
| ... | | | | | | | | | | | | |
| Malignant | 186.00 | 0.772 | 2.360 | 0.044 | 26.00 | 0.443 | -170.00 | -1.356 | -2.326 | -0.101 | 3.405 | 5.952 |
| Malignant | 203.00 | 1.211 | 1.236 | -0.636 | 31.00 | 0.764 | -100.00 | -0.348 | -4.236 | -0.104 | 2.739 | 4.707 |
| Malignant | 198.00 | 1.082 | 3.260 | 0.588 | 20.00 | 0.059 | -190.00 | -1.644 | -4.690 | -0.105 | 3.582 | 6.227 |
| Malignant | 210.00 | 1.392 | 1.260 | -0.622 | 36.00 | 1.085 | -200.00 | -1.788 | -5.236 | -0.105 | 5.653 | 9.347 |
| ... | | | | | | | | | | | | |

Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis,” under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.

# Machine Learning (ML): Medical Image Processing
## "Real-Time Image Processing for Breast Cancer Treatment"

### Real-Time Image Processing for Surgical Procedures



# “High Performance Computing, Machine Learning, and Big Data Analytics for Common Good”

## Outline

QUESTIONS? Any time, please!

- **Introduction**
  - About Myself (Asaduzzaman)
  - Computational Systems: Past, Present, and Future
- **High Performance Computing (HPC)**
  - Hybrid HPC Systems and Parallel Computing/Programming
  - “Regrouping Data/Threads for Improving CPU-GPU Performance”
  - “A Communication-Aware Cache-Controller for HPC Systems”
- **Machine Learning (ML): Medical Image Processing**
  - “Real-Time Image Processing for Breast Cancer Treatment”
- **Geospatial Big Data (BD) Analytics using HPC and ML**
  - “Geospatial Cyberinfrastructure for Common Good”
- **Q/A: Discussion**

# Geospatial Big Data Analytics using HPC and ML

## "Geospatial Cyberinfrastructure for Common Good"

### Problem Statement

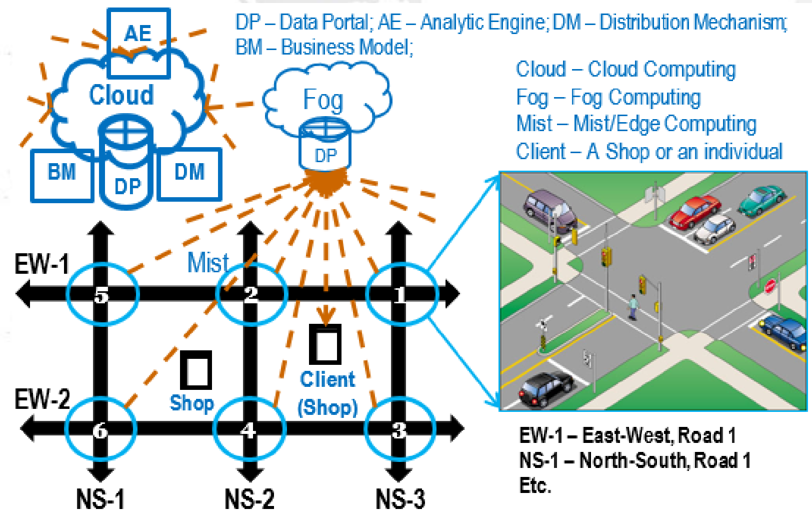
… how the integration of essential geospatial principles (such as spatial constraints in assessing events) with cyberinfrastructure may offer a promising pathway for solving complex problems and improving real-time decision-making practices for economic success …

### Methodology

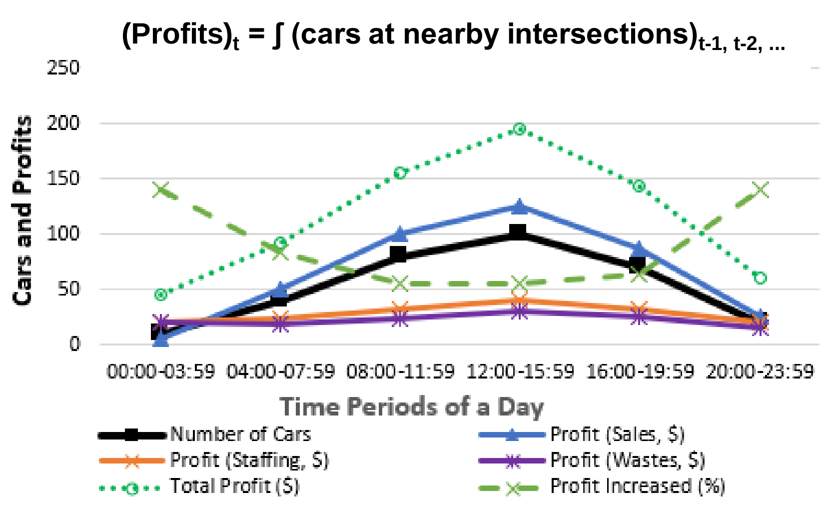
Getting Geospatial Data – Cloud, Fog, and Mist/Edge Computing (HPC)

Analyzing Big Data Faster in Real-Time – Machine Learning (ML)

Make Effective Decisions for Common Good – Big Data (BD) Analytics




An illustration of a geospatial cyberinfrastructure




Traffic information helps make better profits

Asaduzzaman, A., "Geospatial Cyberinfrastructure for Regional Economic Growth and Sustainability," Sustainable Research Pathways Workshop, organized by Sustainable Horizons Institute and Lawrence Berkeley National Laboratory, Berkeley Lab, CA, Dec. 2-3, 2019.

# "High Performance Computing, Machine Learning, and Big Data Analytics for Common Good"

## Outline

**QUESTIONS? Any time, please!**

- **Introduction**
  - About Myself (Asaduzzaman)
  - Computational Systems: Past, Present, and Future
- **High Performance Computing (HPC)**
  - Hybrid HPC Systems and Parallel Computing/Programming
  - "Regrouping Data/Threads for Improving CPU-GPU Performance"
  - "A Communication-Aware Cache-Controller for HPC Systems"
- **Machine Learning (ML): Medical Image Processing**
  - "Real-Time Image Processing for Breast Cancer Treatment"
- **Geospatial Big Data (BD) Analytics using HPC and ML**
  - "Geospatial Cyberinfrastructure for Common Good"
- **Q/A: Discussion**

# WICHITA STATE UNIVERSITY — About WSU

- We are not WSU for WASHINGTON STATE UNIVERSITY . We are .

## ■ What's new…

- New WSU President. Jay Golden, Wichita State's 14th President, starting from January 2020. Golden is a leading researcher in environmental sustainability and an advocate for applied learning and economic development.

## ■ What else…

- Wichita State is the house of the original Pizza Hut. Two Wichita State students, brothers Dan and Frank Carney, started the Pizza Hut business in 1958. The restaurant has since become one of the biggest pizza chains in the world. The original building resides at Wichita State as a museum.
- U.S.News Best Engineering Schools 2020
  #95 Wichita State; #87 wsu.edu;
  #1 MIT; #2 Stanford; #3 UC Berkeley;
  #95 …, KU, UK,

BEST GRAD SCHOOLS U.S.News & WORLD REPORT RANKINGS

**Wichita State University**

Wichita, KS

#95 in **Best Engineering Schools** (tie)

https://www.usnews.com/best-graduate-schools/top-engineering-schools/eng-rankings

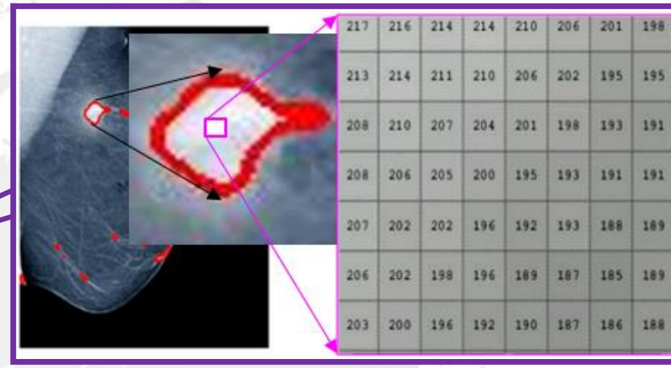
| | |
|---|---|
| Former names | Fairmount College<br>Municipal University of Wichita |
| Type | State university |
| Established | 1895 |
| Affiliation | Kansas Board of Regents |
| Endowment | $247.75 million (2017)[1] |
| President | Andy Tompkins (interim)[2] |
| Provost | Rick Muma |
| Academic staff | 520 |
| Students | 16,058 (Fall 2019)[3] |
| Location | Wichita, Kansas, U.S.[4]<br>37°43′09″N 97°17′35″W |
| Campus | Urban, 330 acres (130 ha) |
| Colors | Black and Shocker Yellow[5] |
| Nickname | Shockers |
| Sporting affiliations | NCAA Division I – The American |
| Mascot | WuShock |
| Website | wichita.edu |

WICHITA STATE UNIVERSITY

CAPP Lab
capplab@wichita.edu

# Research Activities

- Research supported by Kansas NSF, Nvidia, CybertronPC, WSU, …
- Asaduzzaman, A., Sibai, F.N., Mitra, P., Chidella, K.K., Saeed, K.A., and Altaf-Ul-Amin, M., “An Effective Technique to Analyze Poor Contrast Mammogram Images for Breast Cancer Diagnosis,” under review, Elsevier Journal on Expert Systems with Applications (ESWA), Manuscript No. ESWA-D-19-06033.
- Asaduzzaman, A., “Open2C framework and OpenSoC Fabric to build up a communication-aware level-2 cache controller,” 2020 SUMMER RESEARCH AT BERKELEY LAB, Host Scientist John Shalf, Project38, NSA, the DOE Office of Science, and NNSA.
- Sibai, F., El-Moursy, A., and Asaduzzaman, A., “Hardware Acceleration of the STRIKE String Kernel Method for Estimating Protein to Protein Interactions,” under review, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), Manuscript No. TCBB-2020-01-0008.
- Asaduzzaman, A., “A Communication-Aware Cache-Mediator for High Performance Computer Systems to Achieve Energy-Efficient Scalable Performance,” Collaborator Henry Neeman, Director of OSCER at the University of Oklahoma, USA.
- CAPPLab earned top research designation (GPU Research Center) by Nvidia in 2015.



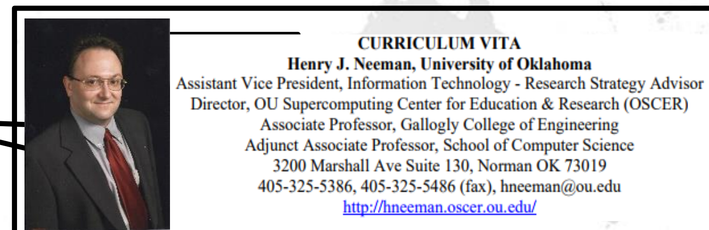



Photo: File

Asaduzzamanâ€™s Computer Architecture and Parallel Program Laboratory (CAPPLab), WSU lab, has been named a GPU (graphics processing unit) Research Center by NVIDIA, the world leader in visual computing.

# Wichita State lab earns top research designation

Monday, November 9, 2015

どうもありがとうございます
(doumo arigatou gozaimasu)

Contact: Abu Asaduzzaman (Zaman)
E-mail: Abu.Asaduzzaman@wichita.edu
Phone: +1-316-978-5261
https://www.wichita.edu/academics/engineering/eecs/faculty/Abu.php

## Q/A: Discussion

**If $x + 1/y = 1$ and $y + 1/z = 1$, what is $xyz$?**