The SyNeRG Lab
http://synergy.cs.vt.edu

The SEEC Center
http://seec.cs.vt.edu

Wu FENG

Dept. of Computer Science

Dept. of Electrical & Computer Engineering → NSF CHREC and Wireless@VT

Health Sciences

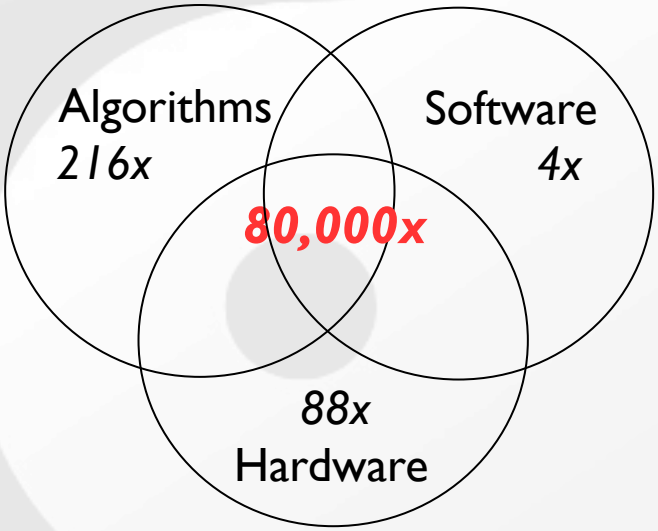
Virginia Bioinformatics Institute

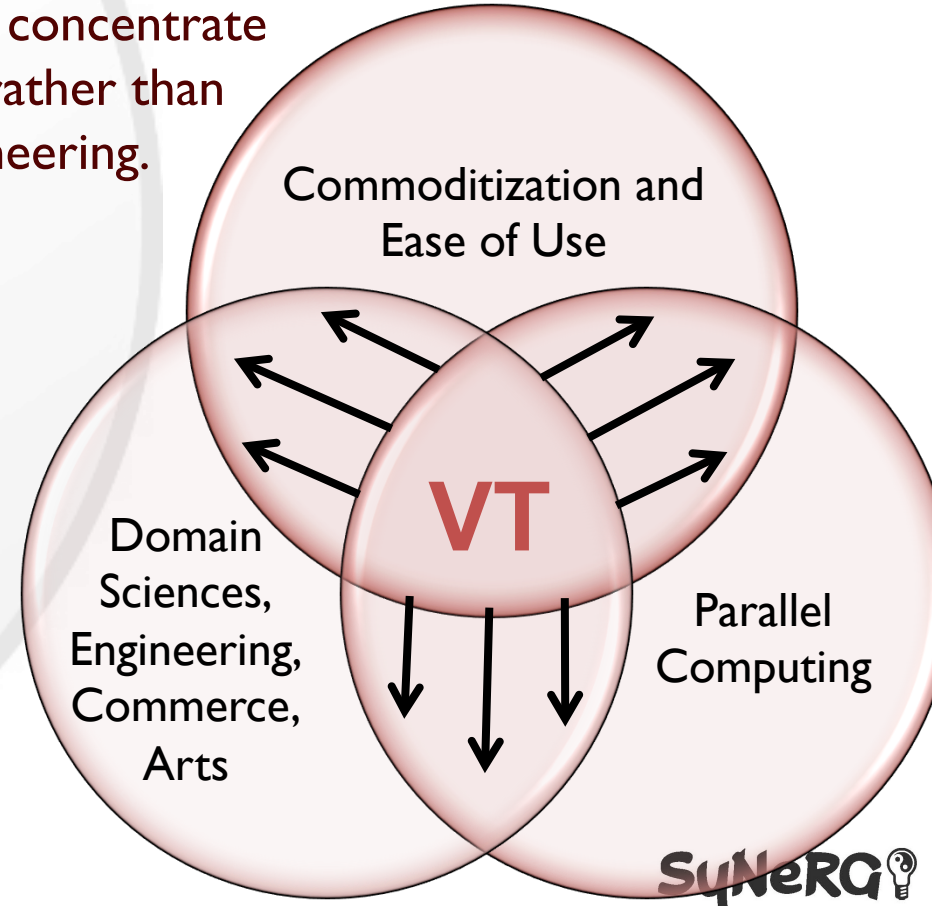
A Little Bit About Me …

- Education
  - Ph.D., Computer Science
    U. Illinois at Urbana-Champaign, 1996
- Professional
  - Current Appointments
    - Professor and Elizabeth & James Turner Fellow; Departments of Computer Science, Electrical & Computer Engineering, and Health Sciences; Virginia Tech
    - Director, SyNeRG Laboratory (http://synergy.cs.vt.edu/) → SEEC Center
    - Founder, The Green500 (http://www.green500.org/)
    - Adjunct Faculty, Virginia Bioinformatics Institute, Virginia Tech
  - Previous Appointments & Professional Stints
    - *Academia:* The Ohio State U. ('00-'03), Purdue U. ('98-'00), U. of Illinois at Urbana-Champaign ('96-'98).
    - *Government:* Los Alamos Nat'l Lab ('98-'06), NASA Ames Research Ctr ('93)
    - *Industry:* IBM T.J. Watson Rsch ('90), Vosaic ('97), Orion Multisystems ('04-'05)

Vision

- Conduct *basic and applied research* in efficient parallel and distributed computing (in the small and the large) via the synergistic co-design of hardware, software, and algorithms
  - Enable scientists and engineers to concentrate on their science and engineering rather than on the *computer* science and engineering.



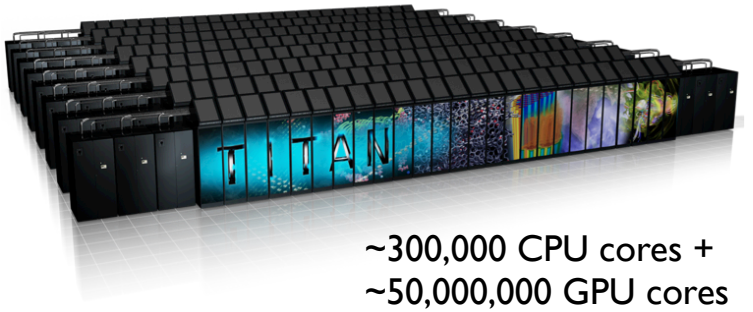


http://www.youtube.com/watch?v=zPBFenYg2Zk

Parallel Computing Across a Spectrum

4 CPU cores +
72 GPU cores

4 CPU cores +
384 GPU cores



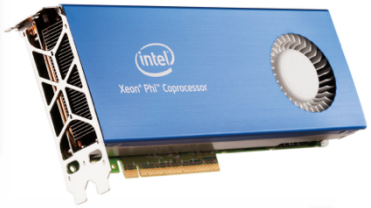
~300,000 CPU cores +
~50,000,000 GPU cores

Our Spectrum of Parallel Computing Resources

Massive parallelism w/ increasing heterogeneity in computing resources

Altera FPGA

AMD APU



Intel Xeon Phi (MIC)

NVIDIA GPU

TI DSP

… across a wide variety of environments

Tianhe-2

Performance via Parallelism

- 2,508 CPU cores
- 187,264 GPU cores
- ~ 6,000,000 threads

Debuted on THE GREEN 500

as *GREENEST* commodity supercomputer in U.S. (11/11)

VirginiaTech
Invent the Future

SyNeRG
synergy.cs.vt.edu

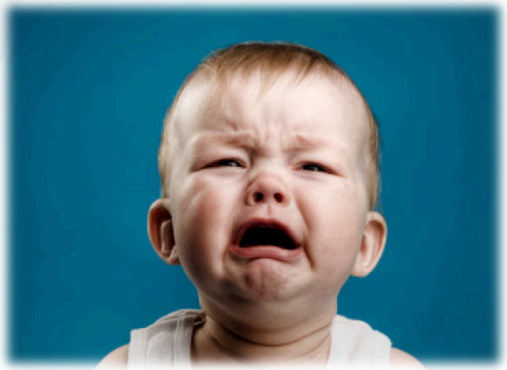
The Challenge

- Vendors delivering parallel computing to the masses
  … at least in hardware
- Many *different* computing environments

Many different architectures

Many different programming languages

Many different optimization strategies



The Challenge: Programmer's View

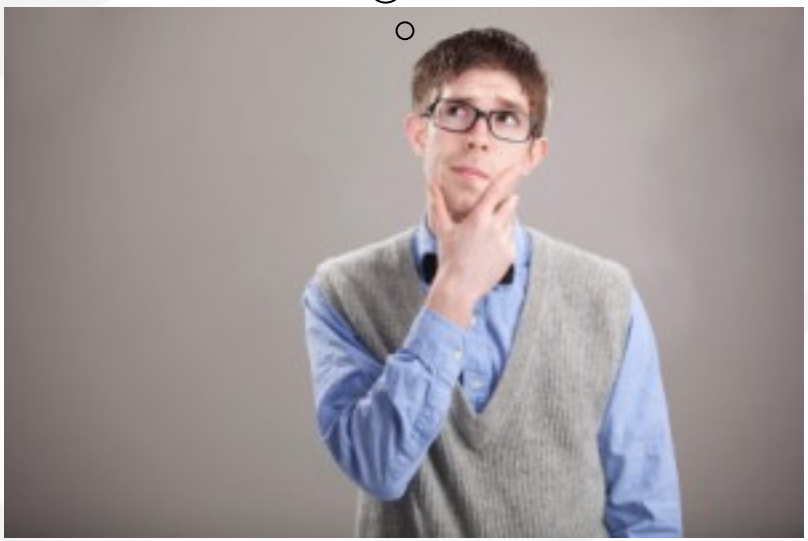
How am I going to program my application for all these platforms? *Do I have to re-program for each platform?*

How do I know which platform to choose to run my application on?

Programmer

The Challenge: Architect's View

How will I evaluate my architecture and compare it to others?



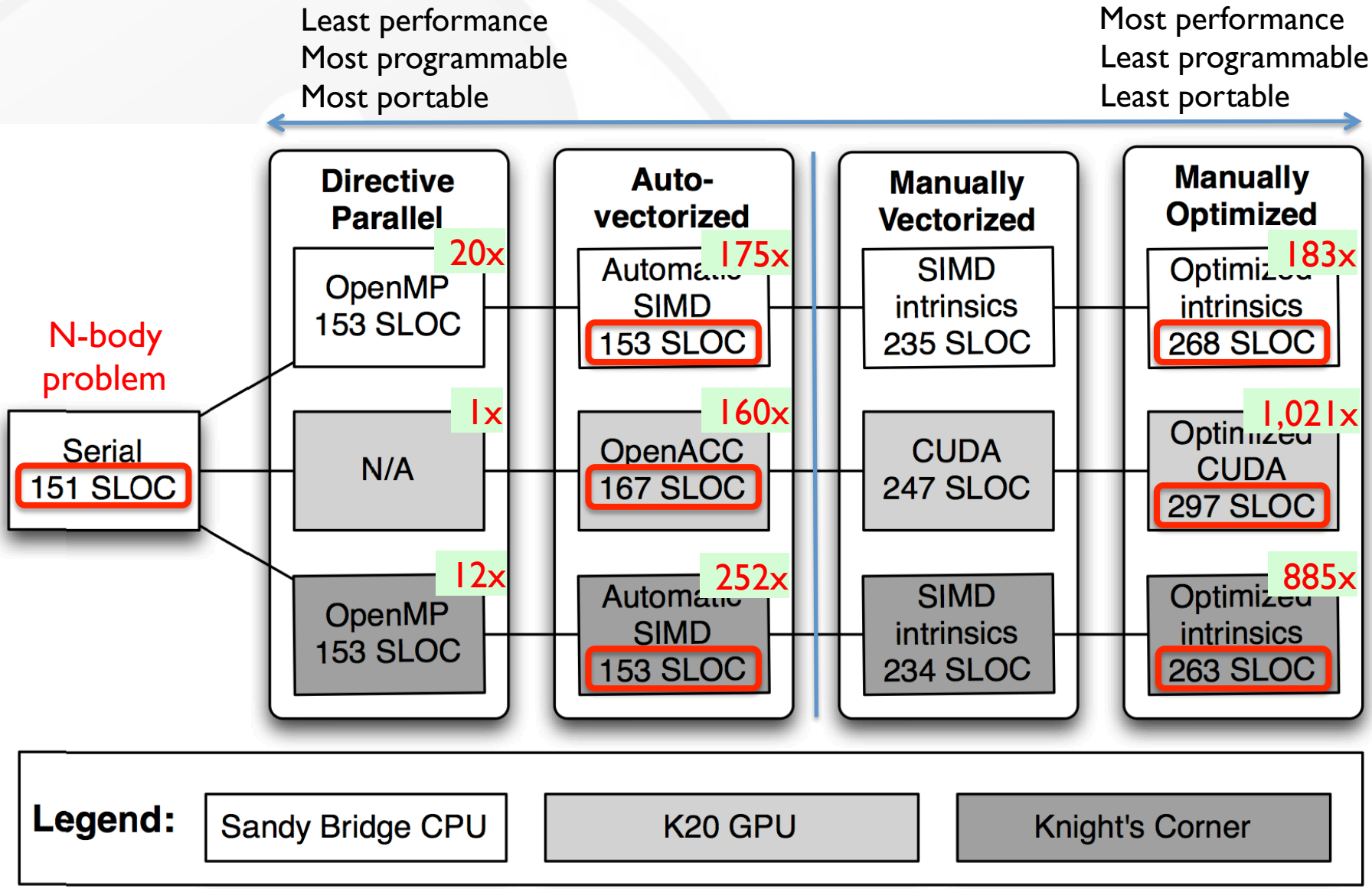
Computer Architect

The Challenge: Compiler & Runtime System

How will I create (automated) back-end optimizations and schedulers and know that they will work well?

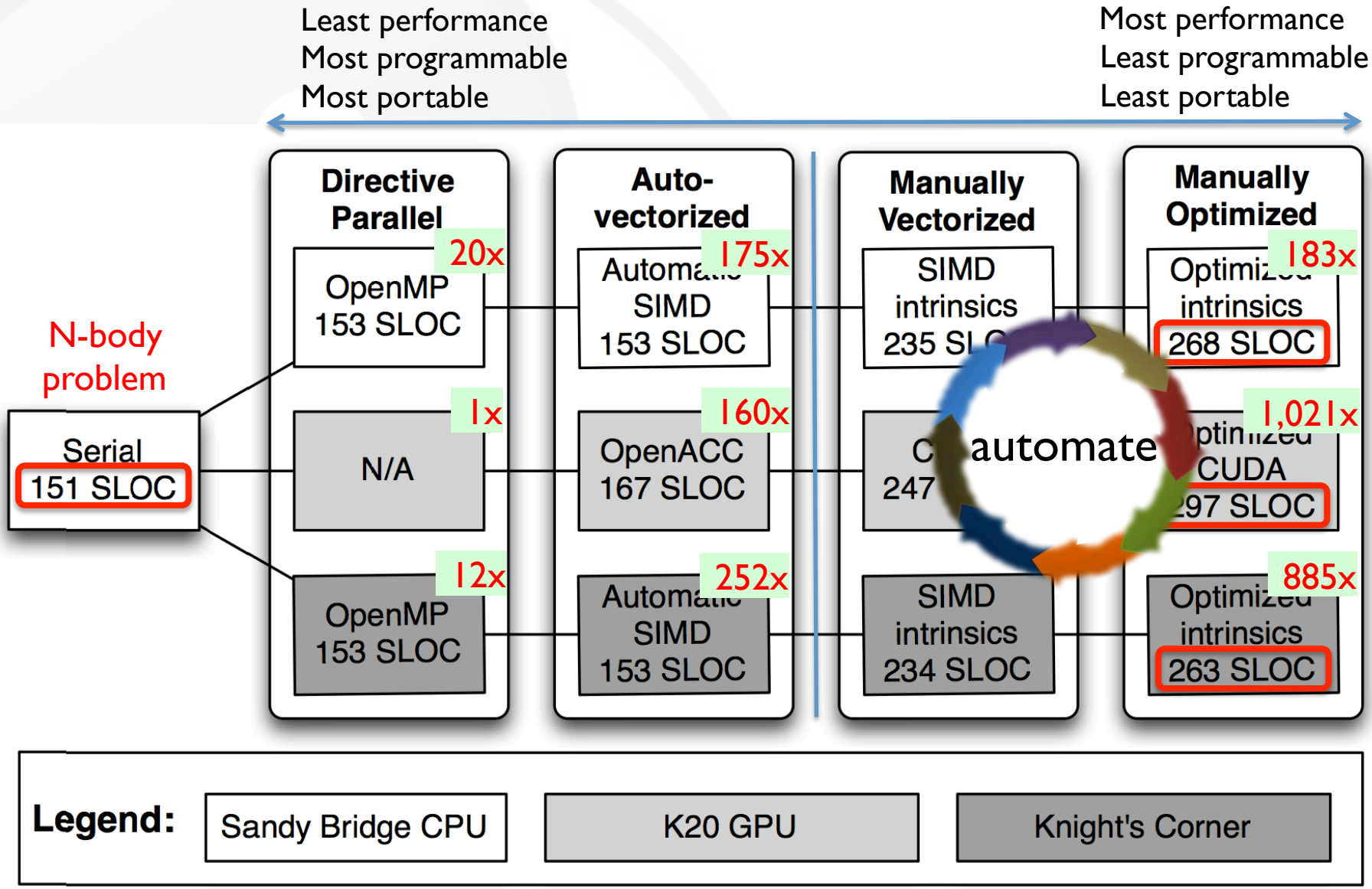
Compiler & Runtime System

"Productivity = Performance + Programmability + Portability"



Applying a similar approach to "Lid-Driven Cavity" and "MetaMorph"

"Productivity = Performance + Programmability + Portability"



Recall: Stanford has ~ 1,000,000 CUDA SLOC that they need to translate to OpenACC/OpenMP or OpenCL

It's More than Co-Design for Performance …

Programmability?

C

```
void MatMul(float* M, float* N,
            float* P, int W) {
  for (int i=0; i<W; ++i)
    for (int j=0; j<W; ++j) {
      for (int k=0; k<W; ++k) {
        P[i*W+j] += M[i*W+k]*
                    N[k*W+j];
      }
  }
}
```

OpenACC

```
void MatMul(float * restrict M,
   float * restrict N,
   float* restrict P,  int W) {
  int i, j, k ;
  #pragma acc kernels
  copyout(P[0:(W*W)]),
  copyin(M[0:  (W*W)],N[0:(W*W)])
  for (i=0; i<W; i++){
    for (j=0; j<W; j++) {
      for (k=0; k<W; k++)
        P[i*W+j]+=M[i*W+k]*
                  N[k*W+j] ;
      }
    }
}
```

CUDA

```
float *d_M, *d_N, *d_P;
int matrix_size=Width*Width*sizeof(float);
cudaMalloc(&d_M, matrix_size);
cudaMemcpy(d_M, M, matrix_size,
           cudaMemcpyHostToDevice);
cudaMalloc(&d_N, matrix_size);
cudaMemcpy(d_N, N, matrix_size,
           cudaMemcpyHostToDevice);
cudaMalloc(&d_P, matrix_size);
dim3 dimGrid(1,1);
dim3 dimBlock(Width,Width);
MatMul<<<dimGrid, dimBlock>>>(d_M, d_
N, d_P,Width);
cudaMemcpy(P,d_P,matrix_size,
           cudaMemcpyDeviceToHost);
cudaFree(d_P);
cudaFree(d_M);
cudaFree(d_N);
```

```
__global__
void MatMul(float* d_M,
            float* d_N,
            float* d_P,
            int W) {
  int row = threadIdx.y;
  int col = threadIdx.x;
  float P_val = 0;
  for (int k = 0; k < W; ++k) {
    float M = d_M[row *W+ k];
    float N = d_N[k *W+ col];
    P_val += M*N;
  }
  d_p[row*W+col] = P_val;
}
```

OpenCL

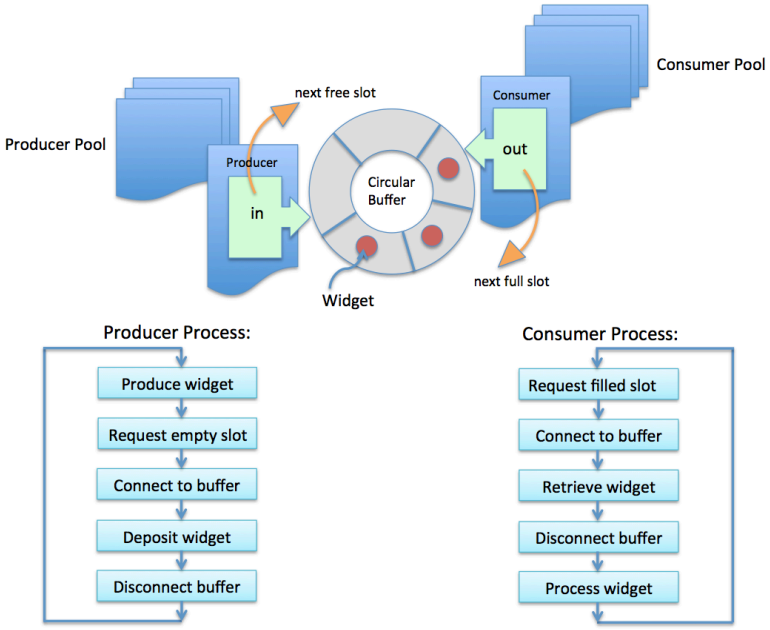
```
/*Code contains parts adapted from code originally written by Tim Mattson
and obtained from: https://github.com/HandsOnOpenCL/Exercises-Solutions/blob/master/Solutions/Exercise08/
*/
char * kernelsource;
cl_int err;
cl_device_id device;
cl_context context;
cl_command_queue commands;
cl_program program;
cl_kernel kernel;
size = W*W;
h_A = (float *)malloc(size*sizeof(float));
h_B = (float *)malloc(size*sizeof(float));
h_C = (float *)malloc(size*sizeof(float));
cl_uint deviceIndex = 0;
parseArguments(argc, argv, &deviceIndex);
cl_device_id devices[MAX_DEVICES];
unsigned numDevices = getDeviceList(devices);
if (deviceIndex >= numDevices)
{
  printf("Invalid device index\n");
  return EXIT_FAILURE;
}
device = devices[deviceIndex];
char name[MAX_INFO_STRING];
getDeviceName(device, name);
printf("\nUsing OpenCL device: %s\n", name);
context = clCreateContext(0, 1, &device, NULL, NULL, &err);
checkError(err, "Creating context");
commands = clCreateCommandQueue(context, device, 0, &err);
checkError(err, "Creating command queue");
d_a = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
                     sizeof(float) * size, h_A, &err);
checkError(err, "Creating buffer d_a");
d_b = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
                     sizeof(float) * size, h_B, &err);
checkError(err, "Creating buffer d_b");
d_c = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
                     sizeof(float) * size, NULL, &err);
checkError(err, "Creating buffer d_c");
kernelsource = getKernelSource("matmul.cl");
program = clCreateProgramWithSource(context, 1, (const char **) &kernelsource, NULL, &err);
checkError(err, "Creating program with matmul.cl");
free(kernelsource);
err = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
if (err != CL_SUCCESS)
{
  size_t len;
  char buffer[2048];
  printf("Error: Failed to build program executable!\n%s\n", err_code(err));
  clGetProgramBuildInfo(program, device, CL_PROGRAM_BUILD_LOG, sizeof(buffer), buffer, &len);
  printf("%s\n", buffer);
  return EXIT_FAILURE;
}
kernel = clCreateKernel(program, "MatMul", &err);
checkError(err, "Creating kernel with matmul.cl");

err =  clSetKernelArg(kernel, 0, sizeof(cl_mem),   &d_a);
       err |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &d_b);
       err |= clSetKernelArg(kernel, 2, sizeof(cl_mem), &d_c);
       err |= clSetKernelArg(kernel, 3, sizeof(int), &W);

checkError(err, "Setting kernel args");
const size_t global[2] = {W,W};
err = clEnqueueNDRangeKernel(
        commands,
        kernel,
        2, NULL,
        global, NULL,
        0, NULL, NULL);
checkError(err, "Enqueueing kernel");
err = clFinish(commands);
checkError(err, "Waiting for kernel to finish");
err = clEnqueueReadBuffer(
        commands, d_c, CL_TRUE, 0,
        sizeof(float) * size, h_C,
        0, NULL, NULL);
checkError(err, "Reading back d_c");
```

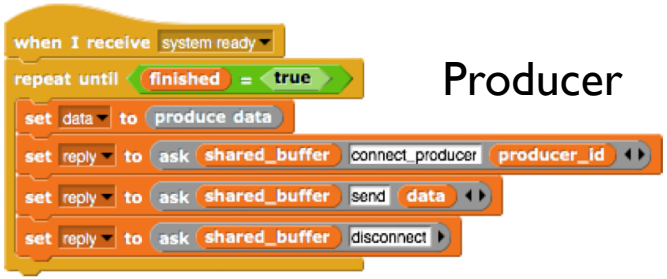
```
__kernel
void MatMul( global float* M,
             global float* N,
             global float* P
             int W) {
int tx=get_global_id(0);
int ty=get_global_id(1);
for(int k=0; k<W; ++k) {
  value+=A[ty*W+k]*B[k*W+tx];
  }
  C[ty*W+tx]=value;
}
```

. . . but portable!

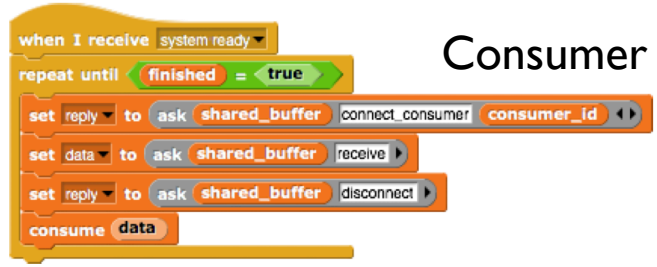
Most Programmable ← → Least Programmable

Parallel Programming with Pictures

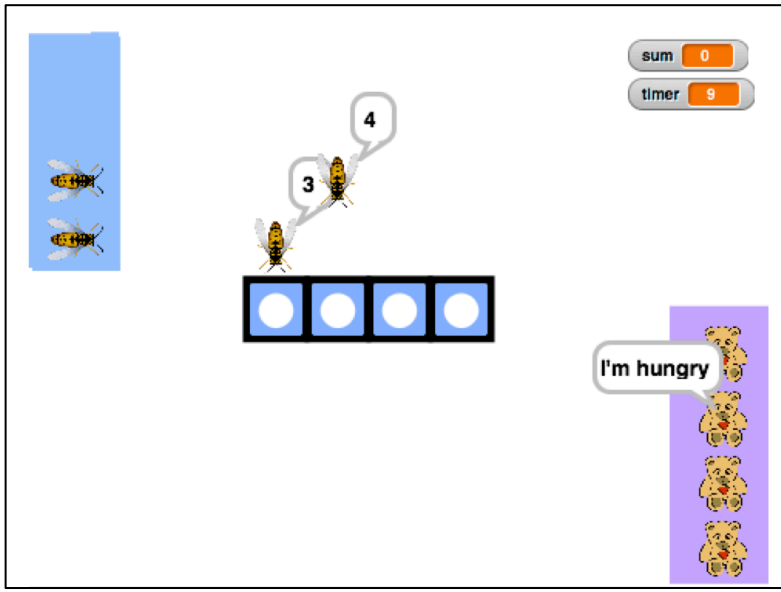


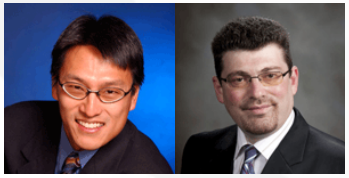


Producer



Consumer



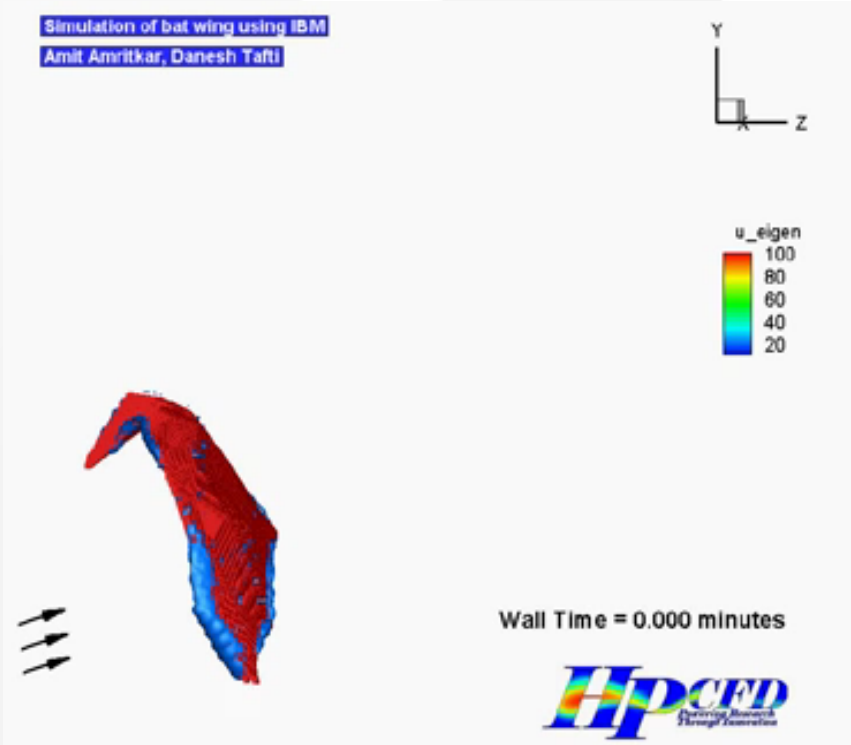


VirginiaTech
Invent the Future

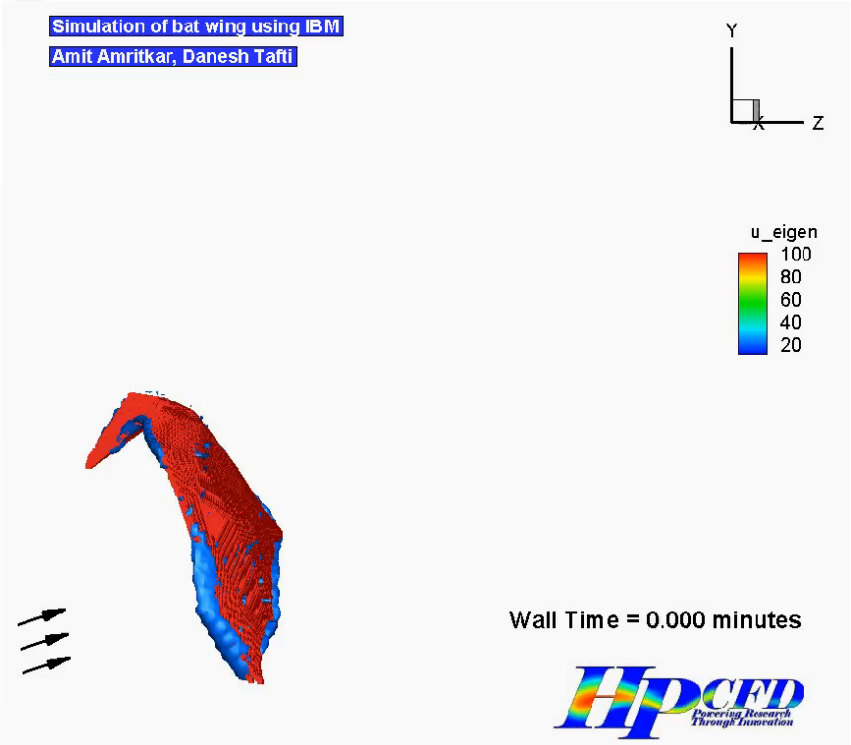
Bat Wing Simulation on CPU and GPU

(Courtesy: Amit Amritkar and Danesh Tafti, Virginia Tech)

CPU



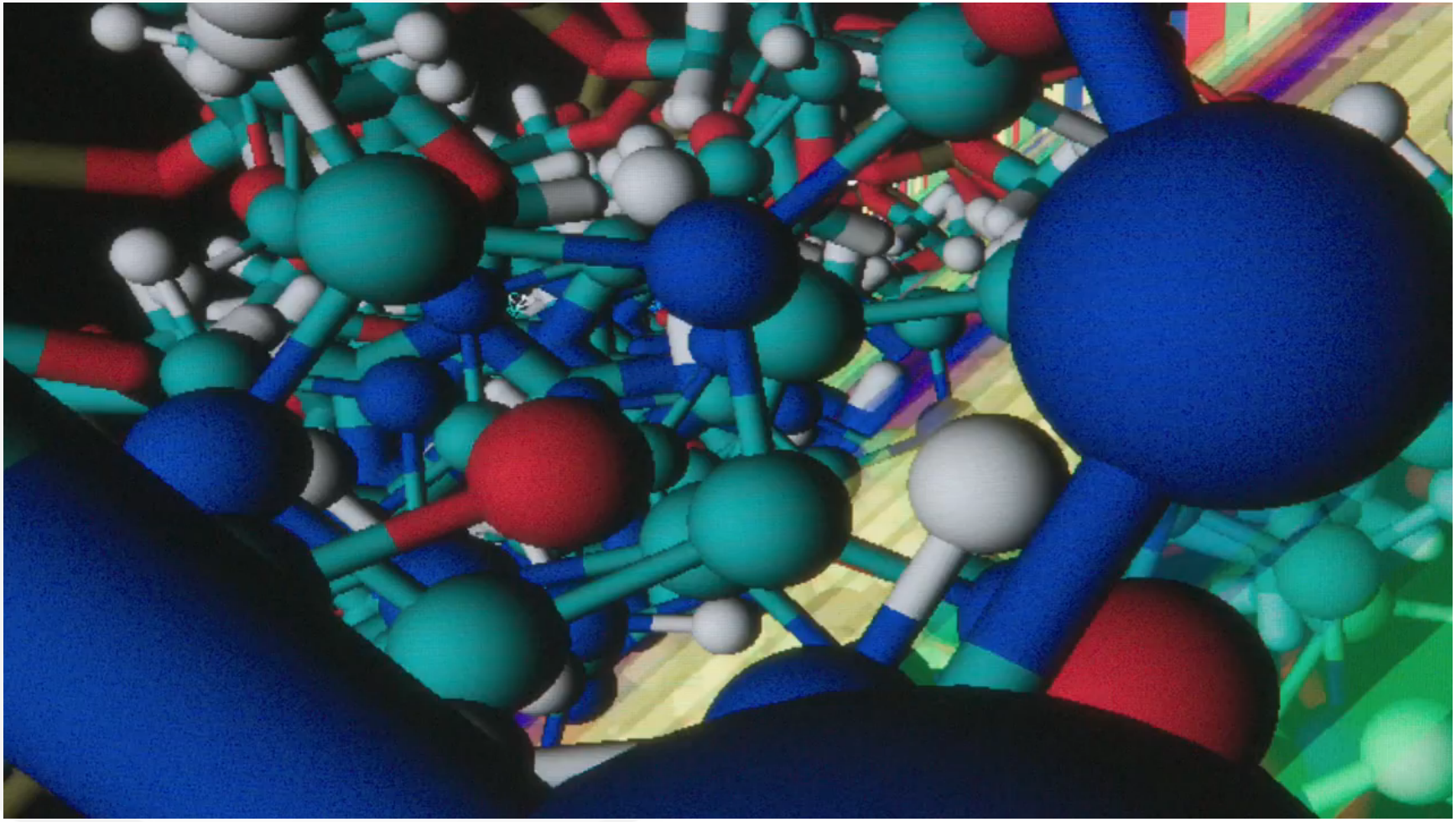
GPU



This work was supported in part by the AFOSR Basic Research Initiative for *Transformational Computing via Co-Design of High-Performance Algorithms & Hardware* via Grant No. FA9550-12-1-0442 and by the GPU-accelerated HokieSpeed supercomputer at Virginia Tech via NSF grant CNS-0960081.

Microsoft Cloud 30-Second Commercial:

Data-Intensive Biocomputing in the Cloud



Microsoft Cloud 90-Second Commercial:

Data-Intensive Biocomputing in the Cloud

Synergy People

Address: Torgersen Hall 2050; 620 Drillfield Drive (Alumni Mall); Blacksburg, VA 24061

Faculty & Staff

Wu-chun ("Wu") Feng
Dept. of CS, ECE, VBI at Virginia Tech and
Dept. of Cancer Biology & Translational Science
Inst. at Wake Forest U.

Mark K. Gardner
Office of IT and Affiliate Faculty
Dept. of CS.

Paul Sathre
Postmasters Research Associate,
Dept. of CS.

Missy Thomas
Administrator for Synergy Lab and SEEC Center.

Nataliya E. Timoshevskaya
Postdoctoral Research Associate,
Virginia Tech.

Harold Trease
Sr. Research Scientist,
Dept. of CS.

Hao Wang
Postdoctoral Research Associate,
Dept. of CS.

Students

Ashwin Aji
(Ph.D.)
NVIDIA Graduate Fellowship

Vignesh Adhinarayanan
(Ph.D.)

Xuewen Cui
(Ph.D.)

L. R. Sriram Chivukula
(M.S.)

Sajal Dash
(Ph.D.)

Islam Harb
(Ph.D.)

Ahmed Helal
(Ph.D.)

Kaixi Hou
(Ph.D.)

Rubasri Kalidas
(M.S.)

Umar Kalim
(Ph.D.)

Konstantinos Krommydas
(Ph.D.)

Sarunya (Kwang) Pumma
(Ph.D.)

Thomas Scogland
(Ph.D.)
NDSEG Fellowship

Balaji Subramaniam
(Ph.D.)

Xiaodong Yu
(Ph.D.)

Da Zhang
(Ph.D.)

Jing Zhang
(Ph.D.)

Collaborators (Current & Past)

Peter Athanas
Dept. of ECE, Virginia Tech.

Pavan Balaji
Argonne National Laboratory.

Keith Bisset
Virginia Bioinformatics Institute,
Virginia Tech.

Eric Brown
Office of IT,
Virginia Tech.

Yong Cao
Dept. of CS,
Virginia Tech.

Bronis de Supinski
Lawrence Livermore

Jack Edwards
Dept. of MAE,

Xiaohui (Helen) Gu
Dept. of CS,

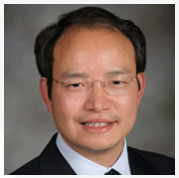
Chung-Hsing Hsu
Oak Ridge National

Hong Luo
Dept. of MAE,

SEEC Faculty and Staff

David
Bevan
BCHM

Ivica
Bukvic
MUS



Guohua
Cao
SBES

Yong
Cao
CS

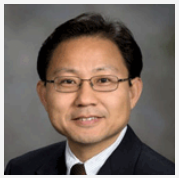
Shengfeng
Cheng
PHYS

T. Daniel
Crawford
CHEM

Eric
de Sturler
MATH

Allan
Dickerman
VBI

Susan
Duncan
FST



Weigui
Fan
ACIS

Wu
Feng
CS/ECE

Daniel
Gallagher
BIOL

Mark
Gardner
CNS, IT

Khidir
Hilu
BIOL

Roderick
Jensen
BIOL

Gregory
Kadlec
FIN

Barbara
Lockee
EDIT

James
McClure
ARC

Kevin
Myles
ENT

Michel
Pleimling
PHYS

B. Aditya
Prakash
CS

Naren
Ramakrishnan
CS

Chris
Roy
AOE

Adrian
Sandu
CS

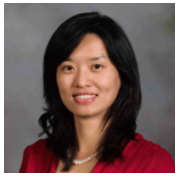
Danesh
Tafti
ME

Eli
Tilevich
CS

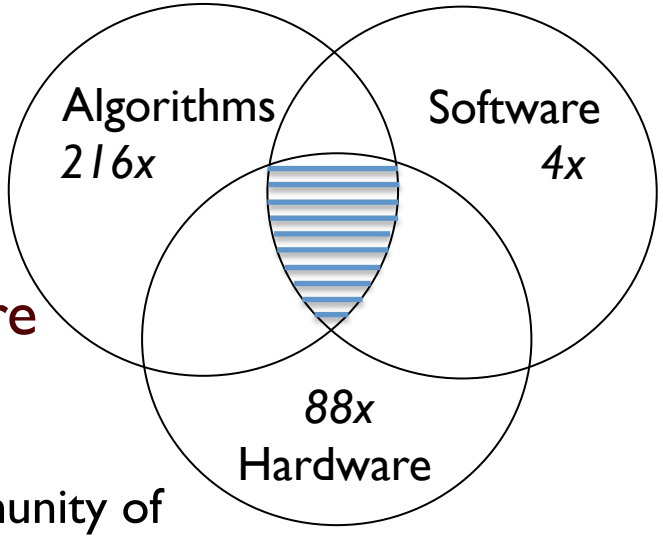
Harold
Trease
ECE

Zhijian
Tu
BCHM

Dane
Webster
Visual Arts



Danfeng
Yao
CS



AFOSR Basic Research Initiative:

Transformational Computing via Co-Design of High-Performance Algorithms and Hardware

This effort is envisioned as a "MURI-like" effort with collaborative grants to teams that will create a new community of researchers skilled in the design, development, and deployment of systems tuned to maximize the synergy of advanced algorithms and high-performance hardware.

New Center at Virginia Tech:

Synergistic Environments for Experimental Computing (SEEC)

VirginiaTech
Invent the Future

Wu Feng, wfeng@vt.edu, 540-231-1192

SyNeRG

http://synergy.cs.vt.edu/

CHREC
NSF Center for High-Performance Reconfigurable Computing

http://www.chrec.org/

mpiBLAST

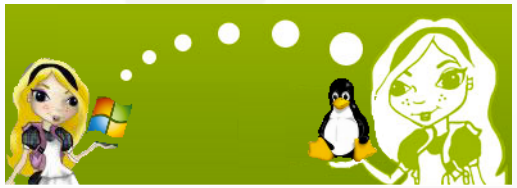
http://www.mpiblast.org/

SUPERCOMPUTING in SMALL SPACES

http://sss.cs.vt.edu/

THE GREEN 500

http://www.green500.org/



http://myvice.cs.vt.edu/

"Accelerators 'R Us"

http://accel.cs.vt.edu/