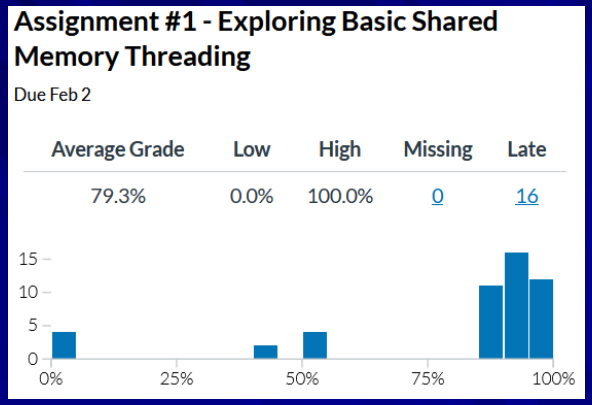
CSCI 551
Numerical and Parallel Programming

Lecture 5-1 – Introduction to MPI and Simulation of Dynamic Systems

February 17, 2025

Assignment #1 & Quiz #1 – Results

- Assignment #1
  - **Ave = 80.05% (a few no attempt made)**
  - Good progress and great start!
  - Keep working on Amdahl's Law (P, SU, S)
  - Methods to make parallel
  - Analysis for speed-up
  - Mr. G's law (saturates htop)
  - Verifies (matches sequential output).
  - Make sure you show results, analyze and explain them well, demonstrate understanding of numerical methods and code
  - Average brought down by "no attempt" – start and turn in what you have done!

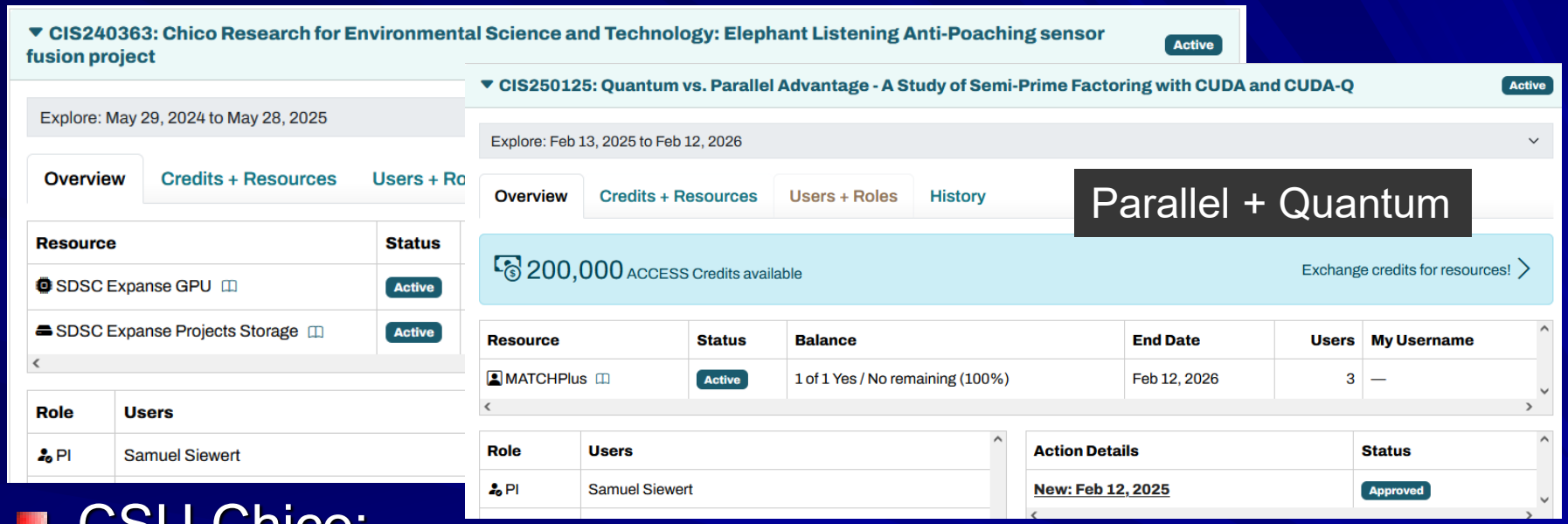


- Quiz #1
  - Quiz #1 **Don't forget to take before mid-night tonight if not already done!**
  - Solutions posted after all have taken
  - Questions on solutions? – Can cover on Thursday
  - Should have basic understanding of Pthreads and OpenMP by now
    - Functional version of OpenMP as well as any well-structured block
    - When to use OpenMP and when to use Pthreads?
  - Vector/Matrix and Prime Numbers – math covered so far

- Grading policy - csci551/policies/CSCI-551-Grading-Breakdown.pdf

Scalable Computing Resources

- NSF ACCESS – Largest scale possible



- CSU Chico:
  - NUC Cluster: 62 nodes - 136 SMT cores
  - ARM Clusters: Jetson Nano cluster (20 nodes, 80 ARM cores, 2,560 SP co-procs), R-Pi-4 cluster (40 nodes, 160 cores)
  - GP-GPU: A100 80GB, 512GB with 16 Xeon cores
  - ESYS: DGX-1 V100 Infiniband cluster (4 nodes).

Exercise #3 – Features MPI

- MPI introduced with Exercise #3
- Read Pacheco, Chapter 3.1 – 3.3 (MPI introduction)
- See MPI examples:
  - csci551/code/hello_cluster/
  - csci551/code/MPI_Examples/
- Make sure you understand ECC-Cluster and CSCI
  - csci551/README-cluster.html
  - See notes on CSCI cluster (Jetson Nano and R-Pi 4)
  - Test hello_cluster on both!
  - Get help from IT or during office hours if you have account issues!

CSCI Cluster and ECC-Cluster Update

- Appears to be working with firewall work-around
- Still testing … works with our starter code with mpiexec
- mpiexec should always work
- mpirun is a bit more complex …

```
sbsiewert@rpi1:~/csci551/numeric-parallel-starter-code/hello_cluster$ mpirun --mca btl_tcp_if_include eth0 -n 8 -host rpi1:4 -ho
st rpi3:4 ./greetings
Hello from process 0 of 8 on rpi1Hello from process 1 of 8 on rpi1
Hello from process 2 of 8 on rpi1
Hello from process 3 of 8 on rpi1
Hello from process 4 of 8 on rpi3
Hello from process 5 of 8 on rpi3
Hello from process 6 of 8 on rpi3
Hello from process 7 of 8 on rpi3
sbsiewert@rpi1:~/csci551/numeric-parallel-starter-code/hello_cluster$
```

- Running with mpirun on ECC-Linux simpler due to difference in firewall and set-up for SSH perhaps

Tips: Don’t Suffer in Silence & Late Work

- Use 3-days grace when life hits-you-over-the-head – do not abuse it (10% penalty for abuse)
- Come to office hours (Sam-Siewert-Fall-2023-Office-Hours.pdf) or make an appointment
  - Don’t rat-hole, e.g., “have over 20 hours sunk into this …”, this is a mistake to not seek out proper help!
  - Get pointed in the right direction
- Asking for help 2 days after assignment due date is not early enough – review assignment when posted, then chip away at it
- We have 2 weeks per assignment so plenty of time to seek out help before the due date including clarifications
- Suffering in silence in industry and noting issues after due dates can get you in trouble. Better to learn now!

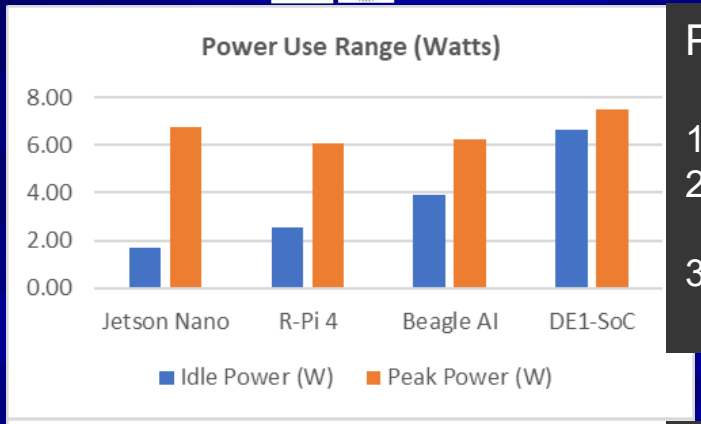
Summary of Image Processing - Scaling

- Work from 2021 by 551 student, ERAU students and myself, submitted to IEEE Aerospace, presentation at Big Sky in March
- Scaling with Power Efficiency (Pthreads, OpenMP, CUDA)
- For Ground, Aerial, and Space Robotics

https://github.com/sbsiewertcsu/CISA-software-benchmarks

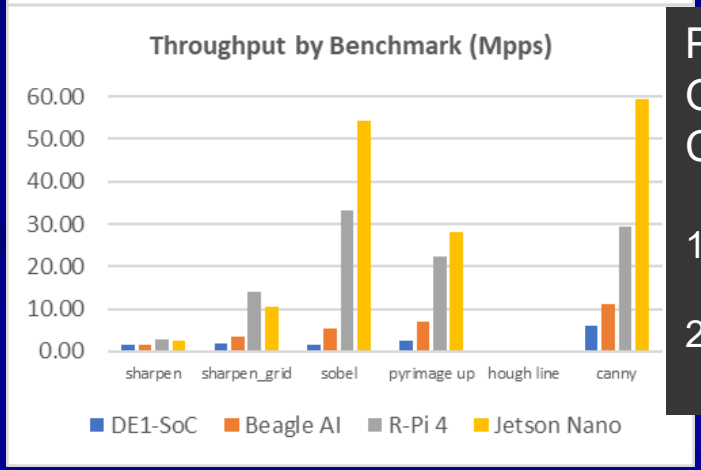
IEEE Aerospace

Student Research Group on THURSDAYS @ 6:30PM PDT, with ZOOM



Power Use:

1. IoT
2. Sensor Networks
3. Edge AI and CV



Pthreads, OpenMP & CUDA

1. Saturate CPU cores
2. Saturate Co-Procs

Terminology Discussion

- What is the difference between?
  - Process (definitely a Container – address space)
  - Thread (at least a Sequence of instructions + what?)
  - Task (in case you are interested)
- Think about examples, address space, processor nodes in clusters, execution context, context switching, sharing data, and sharing code
- Optional Activity: Break-out and discuss
- Report Back: Summarize each

Process:

- Protects apps from rogue apps
  - Bad pointers
  - Stack overflow
  - Non-recoverable exceptions (e.g., x=n/0;)
- Contains at least 1 thread
- Can contain many threads
- Keeps track of resources
  - Open files "lsof"
  - OS mechanisms – e.g., message queue
  - Memory use (heap, stack, data, code, bss)
  - Child processes created (fork, exec)
  - Standard input, output, error
  - Error numbers and state
  - Timers and timeouts
  - Blocked process/thread
  - Signals

Working Definitions

- **Process** – an address space "container" for a thread of execution with stack, register state, and PC state along with significant additional software state such as copies of all I/O descriptors (much more than a task control block for example) **including a protected memory data segment** (protected from writes by other processes). A process may be implemented by a kernel level task (in Linux).

- **Thread (of execution)** - a thread is the execution trace and context of a CPU's PC (Program Counter) over time not including *context switch* code execution by an *RTOS or OS*. State information may or may not be associated with a thread of execution, but the value of the PC before a context switch is the minimum state that must be maintained on a system which includes *preemption, including simple interrupts*.
  - A thread might be as simple as a boot thread, running code out of ROM (Read Only Memory), with no stack, data segment, or anything other than a sequence of machine code instructions.
  - More fully featured user space threads, like Pthreads, have stack, data segments, register state (saved and restored on context switches) and are a method of providing concurrent features without the overhead of a full process in user space and Pthreads share the address space of a process, and are contained within a process.

- **Task** – an "execution context" for a thread with normal thread state including stack, registers, PC (Program Counter), but also including signal handlers, task variables, task ID and name, priority, entry point, and a number of state and inter-task communication data contained in a TCB (Task Control Block). Tasks are the most common execution context used in an RTOS and are also a common OS kernel execution context.

Siewert, Sam, and John Pratt. *Real-Time Embedded Components and Systems with Linux and RTOS*. Mercury Learning and Information, 2015.

MPI – Message Passing Interface

- Software Scaling and Parallel Computation with Clusters
  - Clusters are nodes interconnected via a network
  - Network may range from fully connected to mesh to ring
  - Most modern systems interconnected with a mesh
  - Either a Torus (e.g., Infiniband) or a non-blocking switch (COTS cluster)
  - Non-blocking, line-rate switches for low port count, low data rate, readily available
  - Non-blocking, line-rate switches for high port count, high data rate, are expensive $$$$

16 port, $1,100 on Amazon – 2020
< $100 per port, **10G uplink**
Gigabit < $10 oer port

MPI (Message Passing Interface)

Messages point to point via switch (without going off switch)

Outside connection is simply to get on the cluster

May need to upload/download data sets

MPI – SPMD (pp. 83-113 Pacheco)

- The basic concept is a Single Program that creates multiple data processing processes
  - Processes are created on one node and/or distributed over all nodes in the cluster
  - Messages provide data to each process
  - Each process has a unique identification
  - Each can then work on a portion of the parallel problem
- Results can be merged at the end
- Reduction operators – e.g., summing, maximum, can be applied and create a barrier for ranks

Before Attempting MPI Coding

- Read Instructions - csci551/README-cluster.html

- Check path to required Intel compiler tools
  - /opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/
  - Compiler “mpicc” – intel64/bin/mpicc
  - Cluster tool “mpirun” – /opt/intel/intelpython3/bin/mpirun
  - Single node tool “mpiexc” – /opt/intel/intelpython3/bin/mpiexec
  - C++ Compiler “mpicxx” - intel64/bin/mpicxx
- Make sure you can SSH to each node used without password prompt
  - Otherwise miprun will just HANG
  - Machines being used should all be reachable too
    - csci551/c1_hosts_test.sh
    - csci551/c2_hosts_test.sh

MPI Demonstrations (CSU & Intel)

- MPI greetings with defaults

```
sbsiewert@o244-01:~/code/hello_custer$ ls
c1_machine_file  c2_machine_file  greetings  greetings.c  Makefile
sbsiewert@o244-01:~/code/hello_custer$ mpirun -n 4 ./greetings
Hello from process 0 of 4 on o244-01
Hello from process 1 of 4 on o244-01

Hello from process 2 of 4 on o244-01

Hello from process 3 of 4 on o244-01

sbsiewert@o244-01:~/code/hello_custer$
```

- CSU hello_cluster

```
unzip hello_cluster.zip
cd hello_cluster
make greetings
mpirun -n 4 ./greetings
```

- Intel CSU MPI benchmarks

```
unzip mpi-benchmarks-master-csu.zip
cd mpi-benchmarks-master-csu
make
mpirun -n 4 ./IMB-MPI1
mpirun -n 4 ./IMB-MT
mpirun -n 4 ./IMB-IO
```

```
sbsiewert@o244-01:~/mpi-benchmarks-master$ mpirun -n 16 ./IMB-MPI1
#------------------------------------------------------------
#    Intel(R) MPI Benchmarks 2019 Update 6, MPI-1 part
#------------------------------------------------------------
# Date                  : Tue Sep 22 16:33:58 2020
# Machine               : x86_64
# System                : Linux
# Release               : 4.15.0-117-generic
# Version               : #118-Ubuntu SMP Fri Sep 4 20:02:41 UTC 2020
# MPI Version           : 3.1
# MPI Thread Environment:


# Calling sequence was:

# ./IMB-MPI1

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong
# PingPing
# Sendrecv
# Exchange
# Allreduce
# Reduce
# Reduce_local
# Reduce_scatter
```

Intel Makefile Updated for CSU cluster nodes

https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/mpi-benchmarks-master-csu.zip

OpenMP Demonstrations (Intel CSU)

- csci551/ipsxe_2019_html/compiler_c/openmp_samples/readme.html
- csci551/code/openmp_samples_intel_csu.zip

```
sbsiewert@csci551$ cd openmp_samples_intel_csu/
sbsiewert@openmp_samples_intel_csu$ ls
DCT  Mandelbrot  Matrix_Multiply  MergeSort  readme.html
sbsiewert@openmp_samples_intel_csu$ cd DCT
sbsiewert@DCT$ make run
icpc -c -g -O2 -xAVX -qopenmp -std=c++11  -o release/matrix.o src/matrix.cpp
icpc -c -g -O2 -xAVX -qopenmp -std=c++11  -o release/timer.o src/timer.cpp
icpc -c -g -O2 -xAVX -qopenmp -std=c++11  -o release/DCT.o src/DCT.cpp
TBB Warning: tbb.h contains deprecated functionality. For details, please see Deprecated Features appendix in the TBB re
ference manual.
 Linking...
icpc release/matrix.o release/timer.o release/DCT.o -qopenmp -tbb -o release/DCT
./release/DCT res/nahelam.bmp res/nahelam1.bmp
Please enter the version you want to execute:
0) All versions
1) Scalar version without parallelism
2) OpenMP SIMD without parallelism
3) Scalar version with TBB parallelism
4) OpenMP SIMD with TBB parallelism
4
The time taken in number of ticks is 354533566
The time taken is 0.222091 seconds
sbsiewert@DCT$
```

Intel Threading and OpenMP DCT

```
sbsiewert@Matrix_Multiply$ make run
icc -c -O2 -std=c99 -wd3180 -o out/openmp_sample.o src/openmp_sample.c
icc out/openmp_sample.o  -o out/openmp_samples
Problem size: c(600,2400) = a(600,1200) * b(1200,2400)
Calculating product 20 time(s)

We are using 1 thread(s)

Finished calculations.
Matmul kernel wall clock time = 20.21 sec
Wall clock time/thread = 20.21 sec
MFlops = 3419.477911
sbsiewert@Matrix_Multiply$
```

Intel OpenMP Matrix Multiply

Simple MPI Examples – Build and Run

- Make like any other program, but using mpicc and note special includes (-I) and libraries (-L)

```
2.11 - ecc-linux.csuchico.edu (sbs
sbsiewert@hello_cluster$ ls
c1_hosts            greetings.c        rankscattergather.c  ranksumfan.c          trace.txt
c1_hosts.sam        greetings.cpp      ranksumall.c         ranksumreduce.c
c1_hosts_test.sh    Makefile           ranksumallreduce.c   ranksumtree.c
c2_hosts            piseriesreduce.c   ranksumbutterfly.c   README-cluster.html
c2_hosts_test.sh    piseriessimp.c     ranksum.c            trace2.txt
sbsiewert@hello_cluster$ make greetings
mpicc -g -Wall -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/  -o greetings gr
eetings.c -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib/debug -L/opt/intel/compile
rs_and_libraries_2020.0.166/linux/mpi/intel64/lib
sbsiewert@hello_cluster$
```

- Run with mpirun and specify nodes to use with “-f”

```
sbsiewert@hello_cluster$ mpirun -n 8 -ppn 2 -f c1_hosts ./greetings
Hello from process 0 of 8 on o251-01
Hello from process 1 of 8 on o251-01

Hello from process 2 of 8 on o251-02

Hello from process 3 of 8 on o251-02

Hello from process 4 of 8 on o251-03

Hello from process 5 of 8 on o251-03

Hello from process 6 of 8 on o251-04

Hello from process 7 of 8 on o251-04

sbsiewert@hello_cluster$
```

Follow Steps in:
csci551/README-cluster.html

Test node status:
csci551/c1_hosts_test.sh
csci551/c2_hosts_test.sh

Use: c1_hosts & c2_hosts

Recall Shared vs. Distributed Memory

- Each process is a unique address space
- Each node has a unique memory device

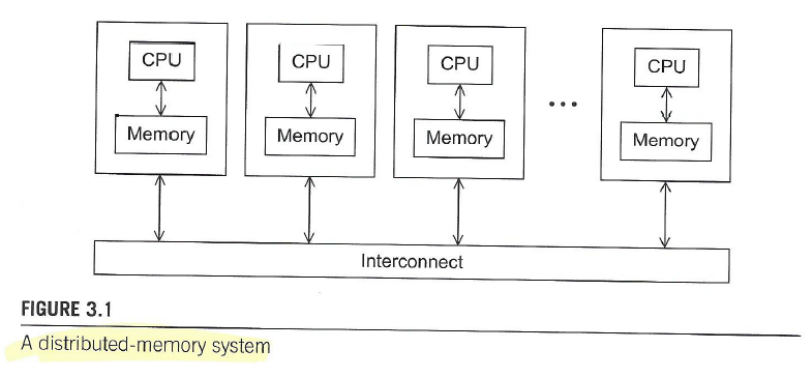


FIGURE 3.1
A distributed-memory system

Interconnect – e.g., switch

CPU + Memory – cluster node

- Shared memory on a single node

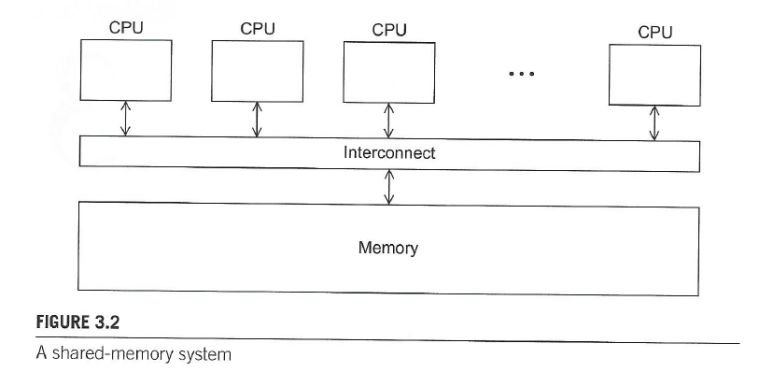


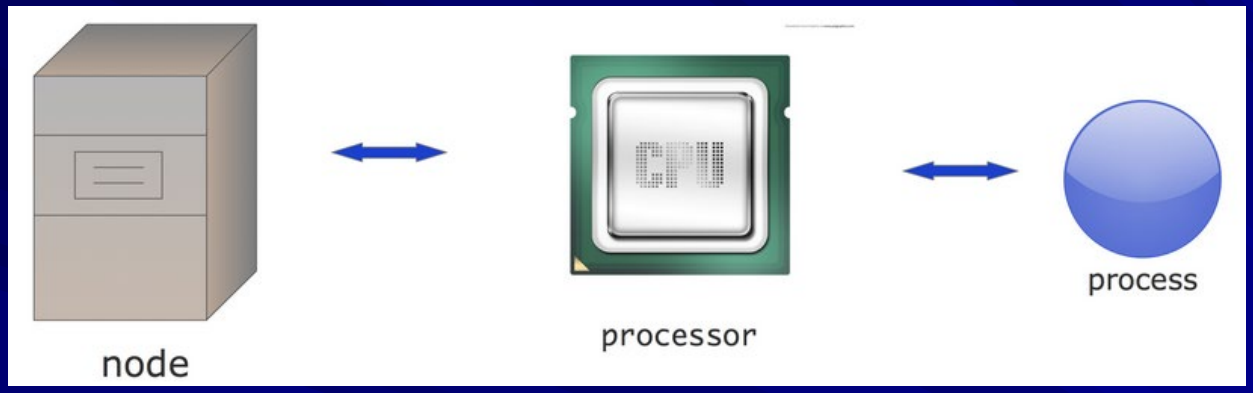
FIGURE 3.2
A shared-memory system

Interconnect
- Memory bus
- Scaling bus (e.g., QPI)

Core + Cache – CPU core
CPU + QPI – CPU socket

Recall Scale-up – Add CPUs with cores

- Scaling up a single node

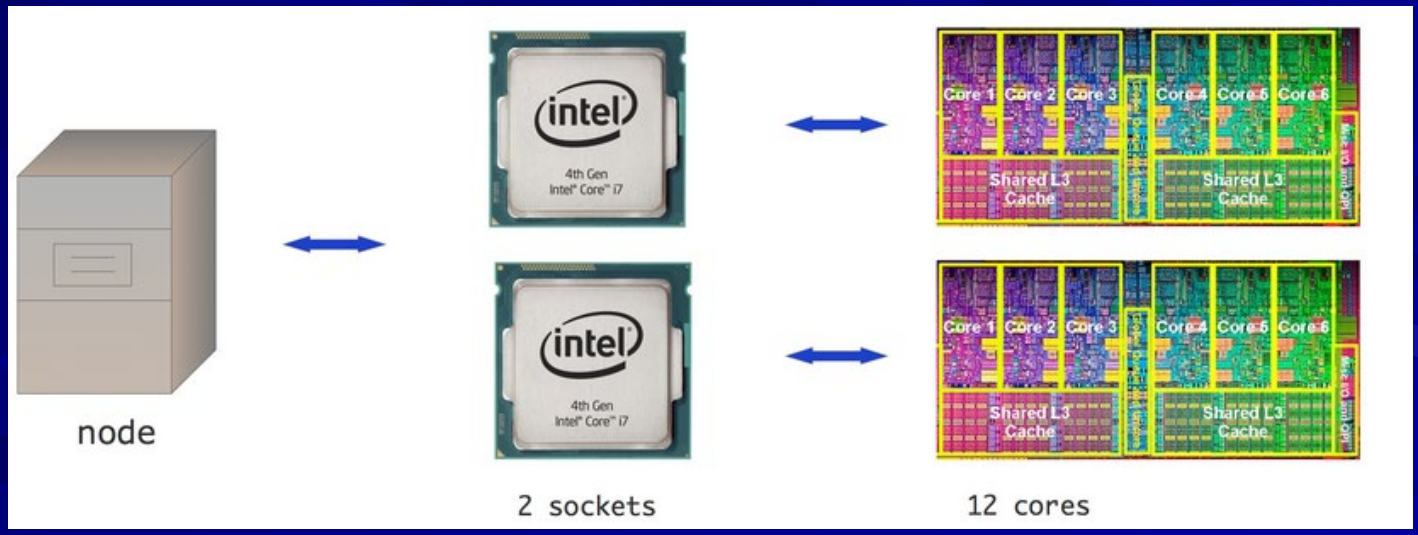


Cost per Core

Intel i7 - $60/core, $30 SMT

ARM R-Pi - $20/core

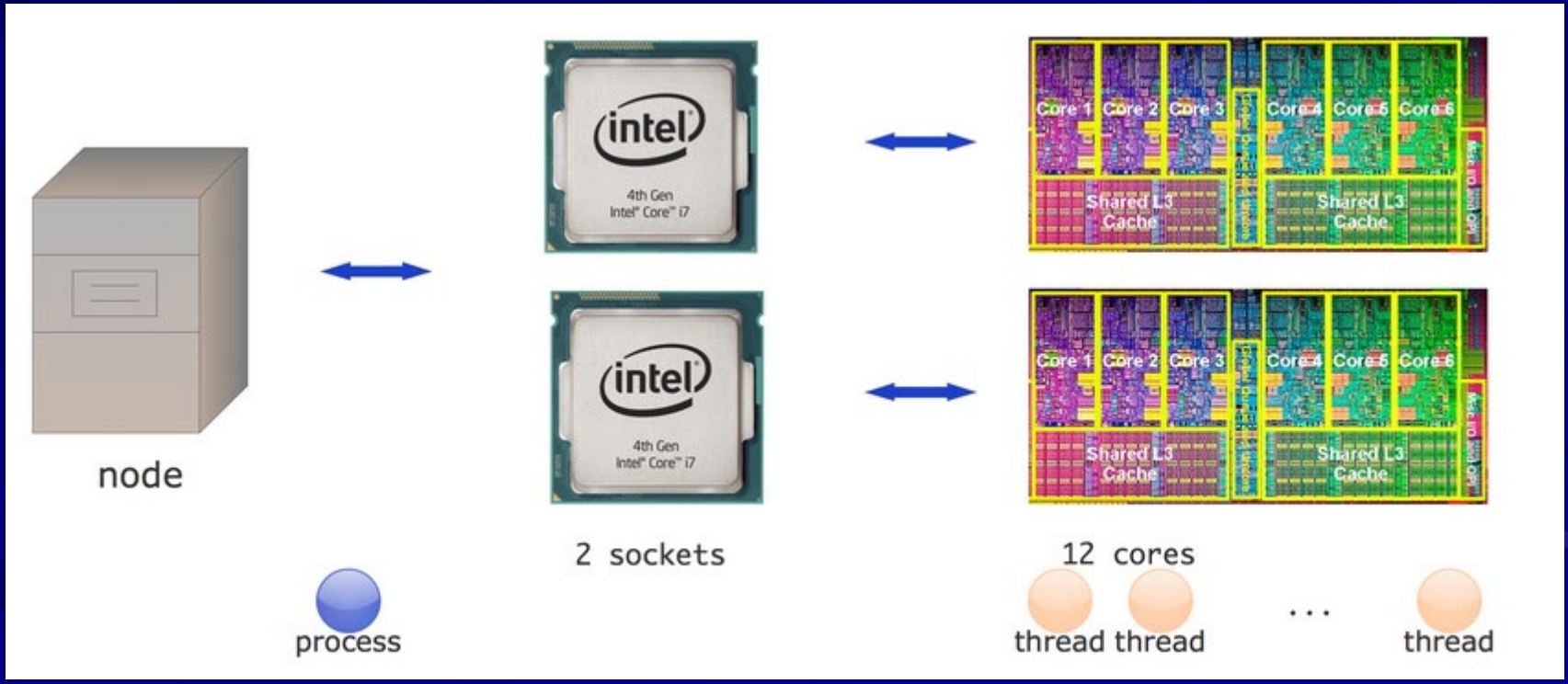
ARM Jetson - $20/core, $1/Co-Proc



https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-functional.html

Recall Scale-up – Shared Memory Pthreading

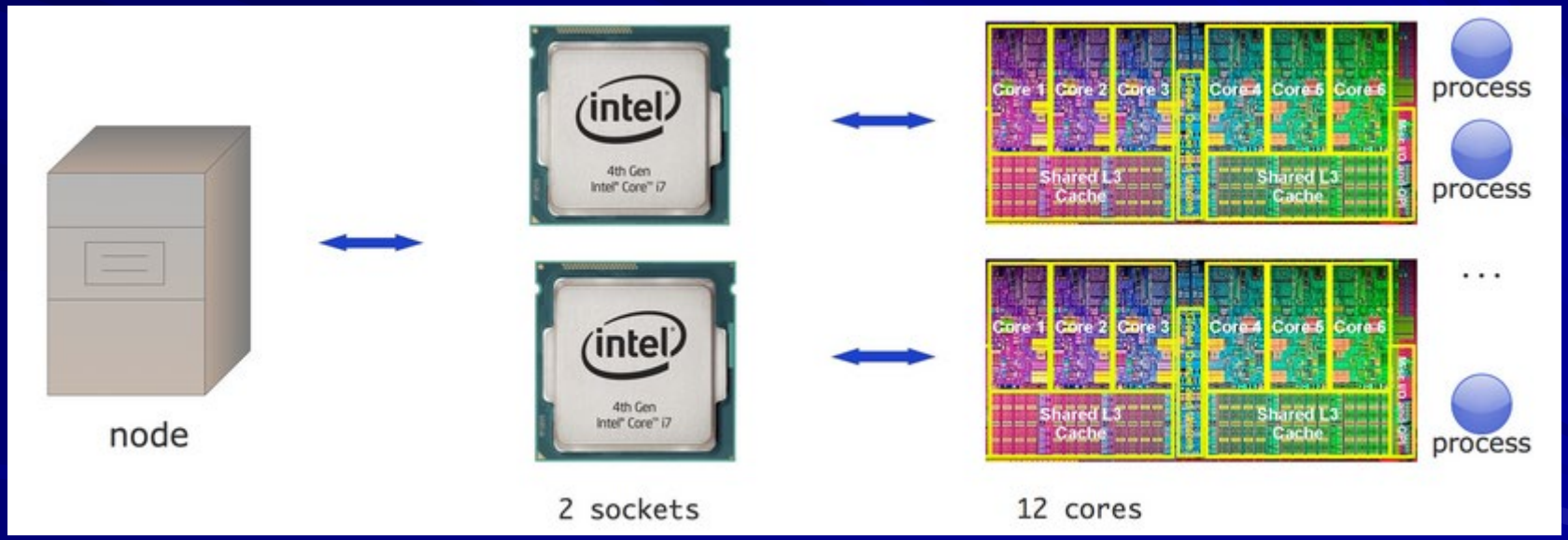
- Scaling up a single node and multi-threading process
  - Shared address space within a single process
  - Threads run on multiple CPUs and processor cores in one address space



https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-functional.html

Recall Scale-up – Multi-programming

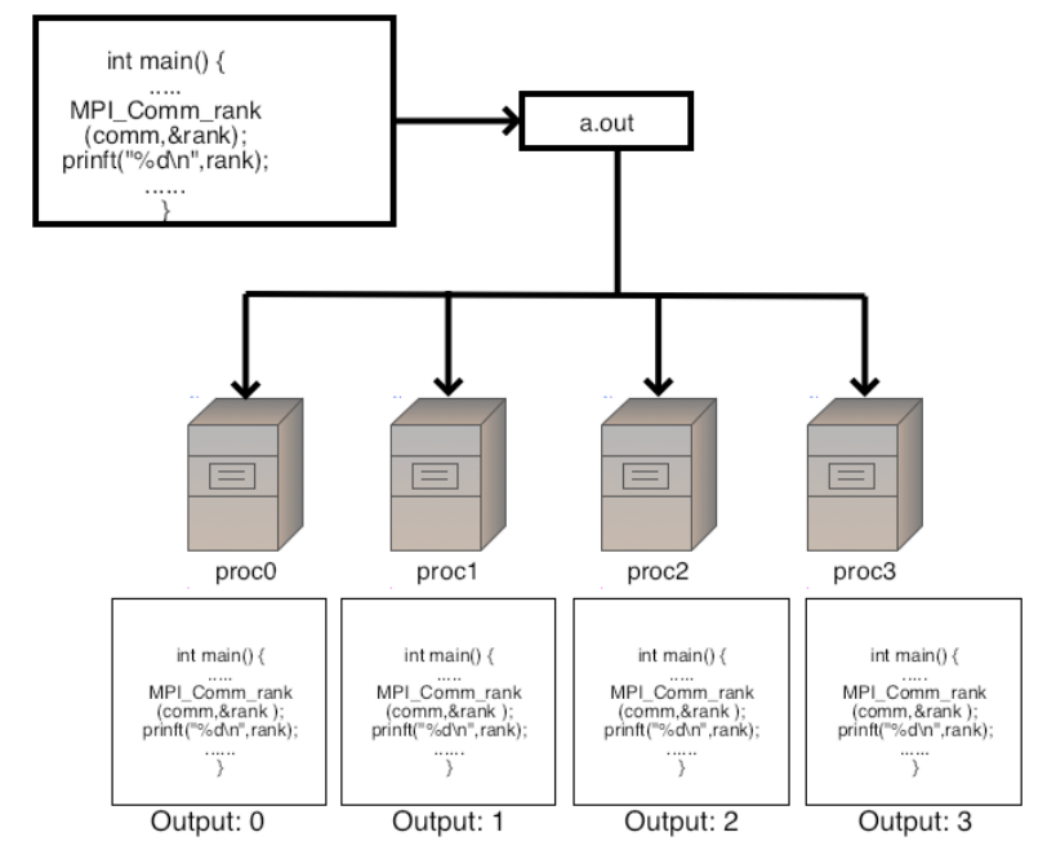
- Scaling up a single node and load balancing processes
  - Separate address space for each process (use messages to communicate and synchronize)
  - Processes run on multiple CPUs and processor cores



https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-functional.html

Scale-out to Multiple Nodes – ECC Cluster

- As I create more processes to divide up work, assign each process to Node, CPU, and core
- Use MPI to divide up work, identify role, merge results



Linux nodes

Each node schedules processes and provides load balancing

SMP UMA/NUMA nodes

Details of node selection and CPU/core selection are provided by system

Cluster + OS

Comparison Discussion

- Which provides most cost-effective and efficient scaling?
  - Distributed memory, cluster scaling with MPI
  - Shared memory, CPU and core scaling with Pthreads or OpenMP
  - Hybrid using both?
- Which is simpler to program?
- Which is easer to use for speed-up?
- Which can be scaled the most?
  - In terms of number of processors and processes/threads

Exercise #3 – Get Started Early!

- **Read chapter 3 through p. 113 – Intro to MPI**
- **Starter - csci551/code/functiongen/**
- **Pacheco Code - csci551/code/MPI_Examples/trap.c**
- Review LLNL tutorials (for more on Pthreads as you are interested)
- Review Intel MPI benchmark user's guide

- Simple MPI program – based on book to explore MPI and cluster features

- Top Down - MPI benchmarks
  - Build, run, observer
  - Review code and explain

- Bottom Up – Methods of integration
  - Simple forward integration, large, medium, small step size
  - Trapezoidal integration, large, medium, small step size

- Application – Train Acceleration and Braking
  - Create a profile of position and velocity over time
  - Given acceleration timeline
  - Compute arrival time
  - Use MPI to speed up the analysis (6 runs, "n" segments to integrate)

Exercise #3 – MPI for Simulation

- Continuous physics of 1-DOF translation
  - Position = P_0 + Velocity x time, if V(t) is constant
  - Velocity = V_0 + Acceleration x time, if A(t) is constant
  - Acceleration can be constant, or could be a function of time

- If Acceleration is f(t), we need to integrate
  - Simple Riemann Sum Integration
  - Position(t) = $P_0 + \sum V(t)dt$
  - Velocity(t) = $V_0 + \sum A(t)dt$
  - Acceleration(t) = f(t), provided as acceleration profile (or from sensor)

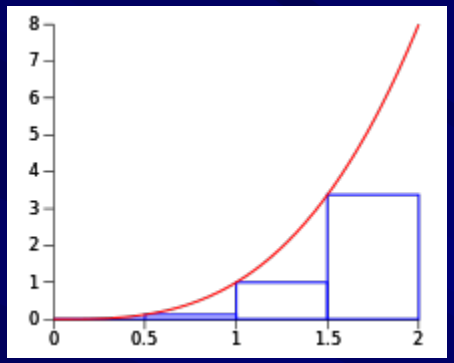
- Approximate the area under curve A(t) for a dt (step size)
- Break A(t) into segments or make it a look-up-table
- Interpolate for A(t) at times between profile given
- Consider accuracy (method of integration & step size) and precision (double or float)

Methods of Integration (Riemann Sum)

Left Riemann Sum (Forward)

A_n= f(a) dx

dx=[b-a]/n

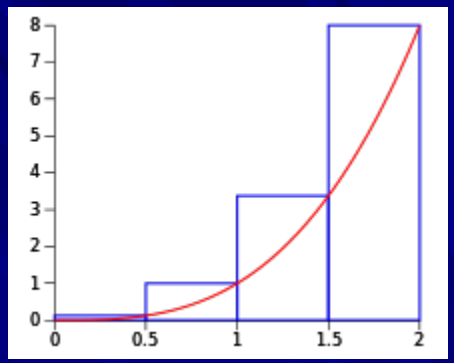


A_{b-a} = dx [(f(a) + f(a+dx) + f(a+2dx) + … + f(b-dx)]

Right Riemann Sum (Backward)

A_n= f(a+dx) dx

dx=[b-a]/n

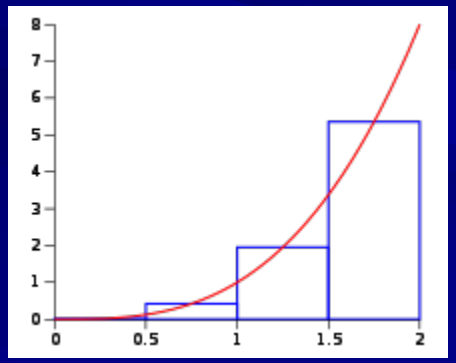


A_{b-a} = dx [f(a+dx) + f(a+2dx) + … + f(b)]

Midpoint Rule

A_n= f(a+dx/2) dx

dx=[b-a]/n



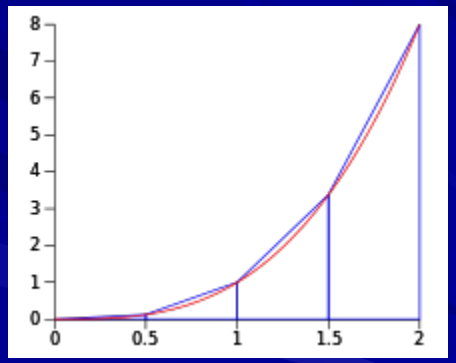
A_{b-a} = dx [f(a+dx/2) + f(a+3dx/2) + … + f(b-dx/2)]

Trapezoidal Rule

A_{trap}=½ h (f(a)+f(b))

dx=[b-a]/n

A_n=½ dx (f(a)+ f(a+dx))



½ dx [(f(a) + f(a+dx)) + (f(a+dx) + f(a+2dx)) …]

½ dx [f(a) + 2f(a+dx) + 2f(a+2dx) + … f(b)]

Starter code: https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/hello_integrators/

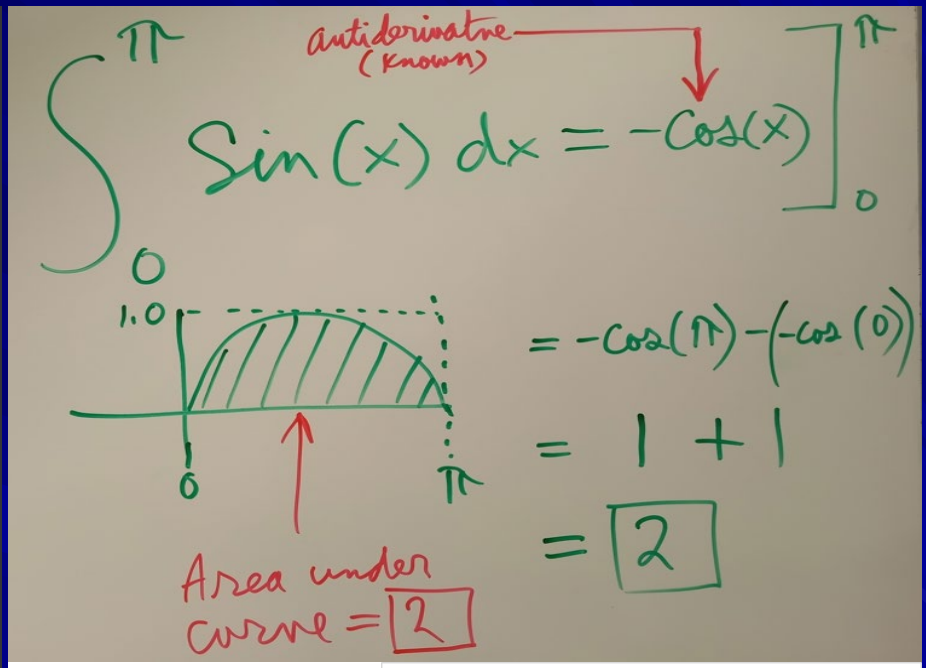
How do I verify Numerical Integration?

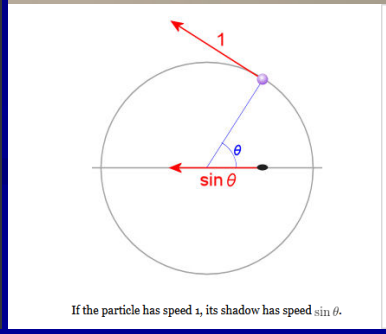
- Test with definite integral from Calculus – exact solution
- E.g., area under Sin(x) over well-defined interval (2.0)

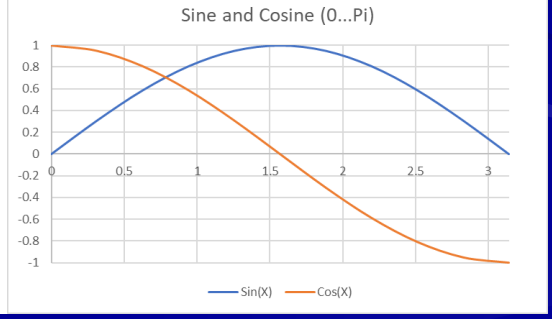
Known anti-derivative of sine and evaluation
- Definite integral solution
- We know from calculus
- Can be proven

Use this fact to verify your numerical integration
- Riemann Sum (function)
- Trapezoidal (function)
- Simpson's Rule (function)
- Runge-Kutta (differential equations)
- Other methods of quadrature (math topic)
- Estimation of area under a curve (function)
- Integration in general of a differential equation
- In applications, the functions generally represent physical quantities, the derivatives represent their rates of change, and the differential equation defines a relationship between the two.

Unit Circle integration of velocity of shadow
- Assume light source on vertical axis
- Velocity of the shadow is sine(theta)
- Position is the Integral of velocity
- Distance is 2.0 for unit circle

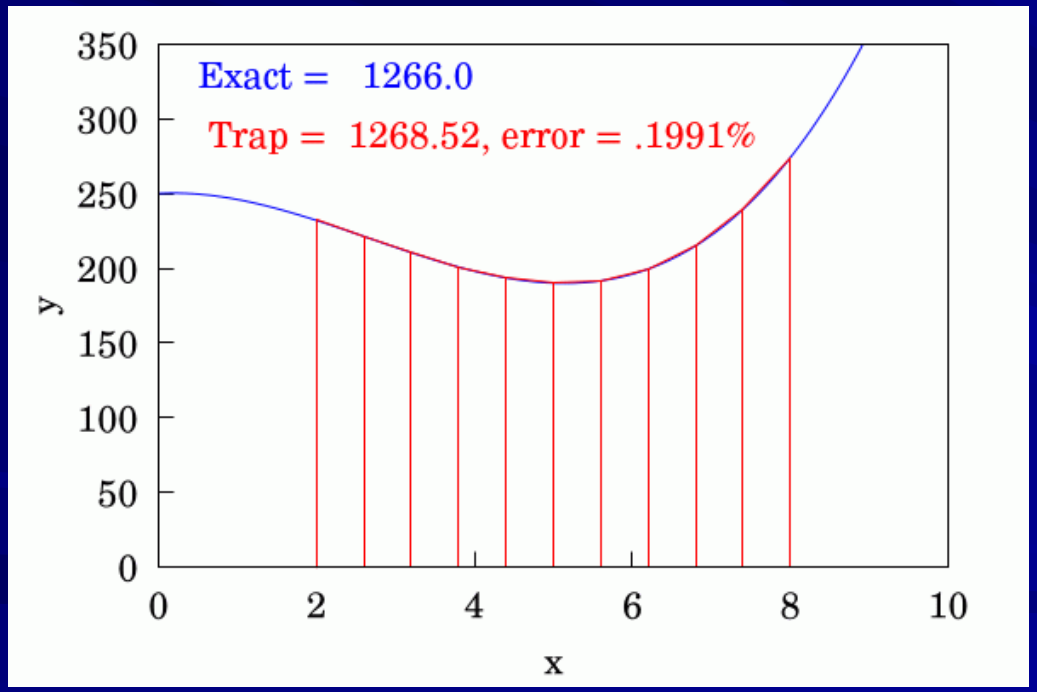






Trapezoidal Rule

- Consider following for error in numerical integration
  - Step size (smaller can reduce error)
  - Method of estimating area over interval (can be more accurate)
  - Difficult to prove one method is always best – depends on function, increases complexity of code, could be fixed or variable



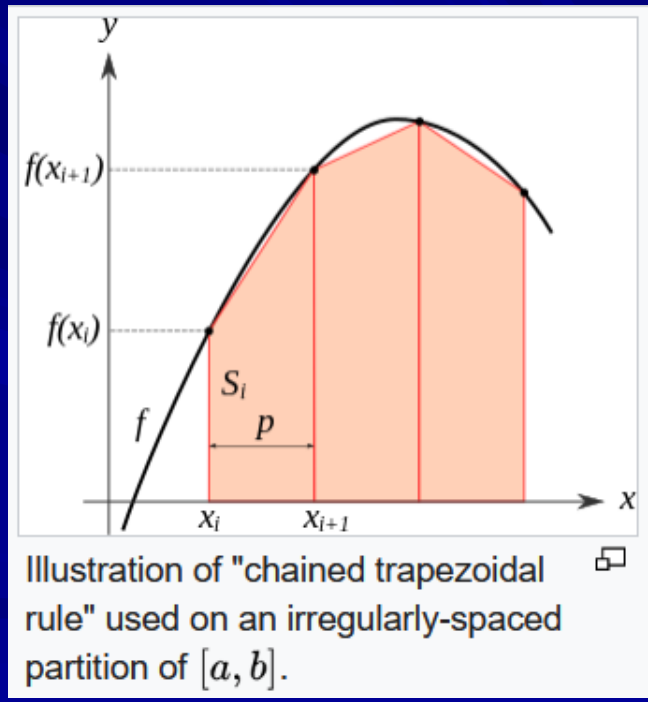
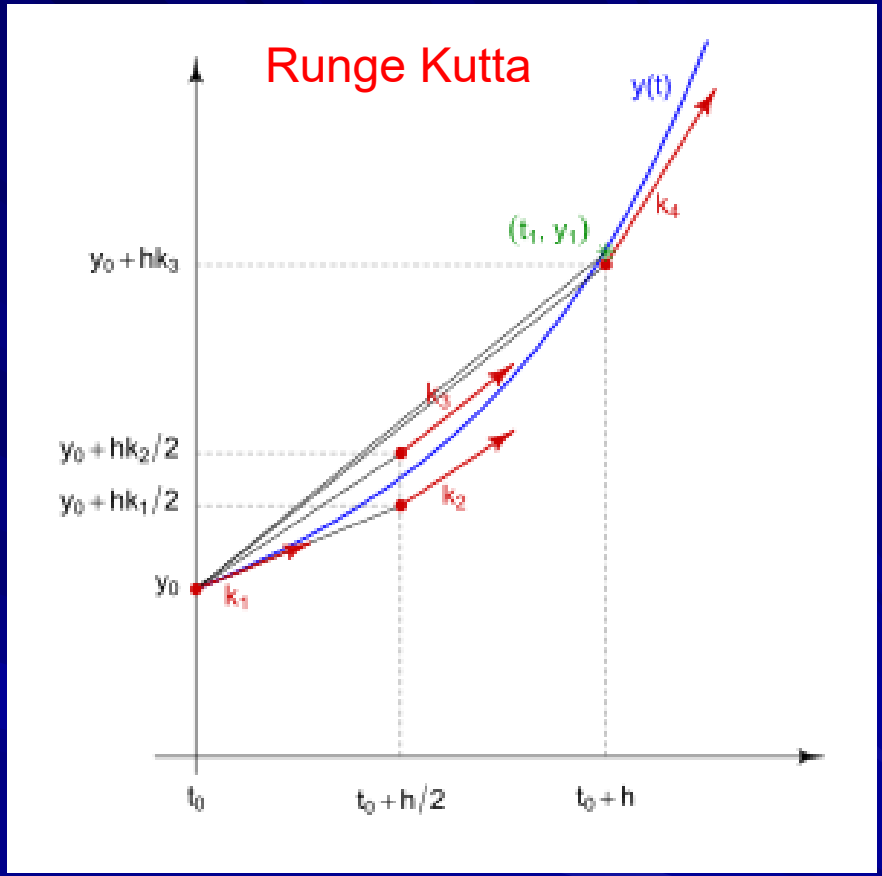


Illustration of "chained trapezoidal rule" used on an irregularly-spaced partition of $[a, b]$.

https://en.wikipedia.org/wiki/Trapezoidal_rule

More Advanced Methods

- Trapezoidal Rule (functions)
  - Can be better than Riemann for same step size
- Simpson's Rule (functions)
  - Can be better than trapezoidal for same step size
- Runge-Kutta methods (differential equations)
  - Fixed step size, 4 increments for RK4
  - We are estimating 4 slopes over the interval
  - We are averaging k1, k2, k3 and k4 and weighting the k2 and k3 by 2x



$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

Summary – Numerical Integration

- Use Simple Riemann Sums – Left, Right, Mid-point
- Consider Trapezoidal and Simpson's
- Step size matters!
  - Smaller is generally better, but requires more computation
  - Think of step size as you do "resolution" for images
- Advanced methods might allow for larger step sizes or even variable step sizes
- Differential and difference equations (for the curious) are interesting for more advanced
  - e.g., Lorenz Equations, N-body simulation
  - Has y and dy/dt in same equation (can have d^2y/dt for second order)
- Simple integration useful for any problem that involves the area under a curve (function), e.g., our train problems
- Equations of motion – can be simple integration problem (e.g., our 1-DOF train problem), or more complex differential equations depending on system (e.g., spring mass damper system)