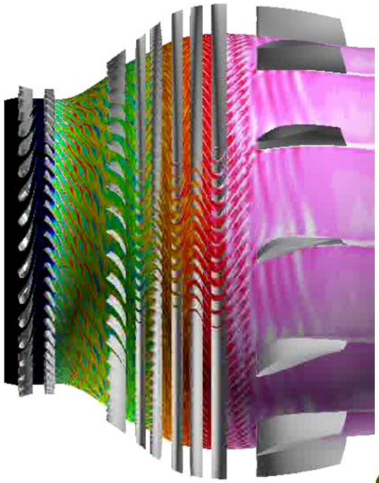
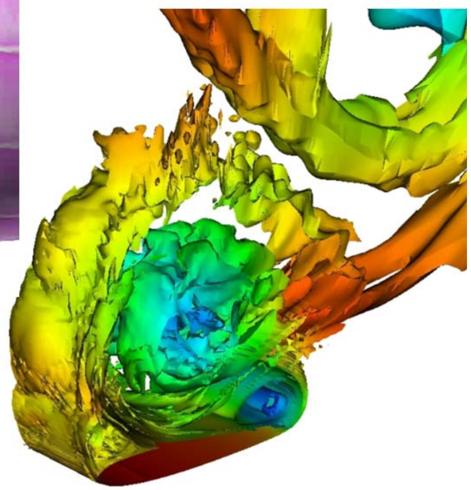
Parallel Computations in Fluid/Thermal Sciences

- **Welcome to MAE267!**

- **My name is Roger L. Davis**
  - 2104 Bainer Hall
  - Phone 530-752-2264
  - davisrl@ucdavis.edu
  - Office Hours by appointment





- **This purpose of this course is to teach engineers how to develop engineering software using parallel computers.**

Grades

- **Grades will be based upon**
  - 20% on homework
  - 80% on projects

- **Schedule of homework, lecture notes, etc. on smartsite**

- **Classes will consist of lectures, programming workshops, and discussion of homework/projects**

- **Lecture notes, etc. can be downloaded from web at:**

smartsite.ucdavis.edu

References

- **Here are some references used for this course:**
  - "UNIX In a Nutshell," Arnold Robbins, 3rd Edition, O'Reilly
  - "Introduction to Parallel Computing", Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, Second Edition, Addison Wesley
  - Introduction to Fortran 90/95," Stephen J. Chapman, First Edition, WCB-McGraw Hill
  - "Using MPI – Portable Programming with the Message-Passing Interface," William Gropp, Ewing Lusk, and Anthony Skjellum, Second Edition, MIT Press
  - "Parallel Programming with MPI," Peter S. Pacheco, Morgan Kaufmann Publishers, Inc.
  - "Numerical Linear Algebra for High-Performance Computers," Jack Dongarra, Iain Duff, Danny Sorensen, and Henk van der Vorst

References (Cont)

- **Here are some more references used for this course:**
  - “Parallel Methods in Numerical Analysis,” Stanford University
  - “Computer Architecture – A Quantitative Approach” John L Hennessy and David A Patterson, Second Edition, Morgan Kaufmann
  - “Sourcebook of Parallel Computing”, edited by Jack Dongarra, Ian Foster, William Gropp, Ken Kennedy, Linda Torczon, and Andy White, Morgan Kaufmann
  - “Numerical Recipes in Fortran 90 – The Art of Parallel Scientific Computing” William Press, Saul Teukolsky, William Vetterling, and Brian Flannery
  - “Object-Oriented Programmin via Fortran90/95”, Ed Akin, Cambridge University Press, 2008

Overview of Course

- **Introduction**
  - Overview of Course, Engineering problem types that must consider parallel computing
- **Overview of Fortran 90/95 for engineering programs**
  - structure of statements and programs, assignment statements, intrinsic functions, I/O, branches and loops, arrays, modules, data-types, pointers, memory allocation
- **Parallel Computer Architectures**
  - vector processors, SMPs, distributed-memory, Beowulf clusters, advantages/disadvantages
- **Parallel Performance Models and Analysis**
  - bandwidth, latency, speedup, Amdahl's law, performance analysis tools
- **MPI (distributed-memory) vs OpenMP (shared-memory) computers and programs**
- **Overview of Data Structures**
  - multi-block structured, unstructured, hybrid, mesh refinement, implicit and explicit algorithms

Overview of Course (cont)

- **Shared memory programming and computing with cores**
- **Accelerators using graphical processing units (GPUs) or Intel Xeon Phis**
- **Domain decomposition**
  - graph partitioning, bisection, Metis, ParMetis, Chaco
- **Distributed memory programming**
  - message passing interface, MPI
  - message passing interface, issues with multi-block structured solvers
  - message passing interface, issues with unstructured-grid solvers
- **SPMD vs MPMD programming**
  - considerations for multi-disciplinary engineering simulations
- **Other parallel engineering applications – optimization and sorting**
- **Impact of parallel computing on implicit and explicit solution algorithms, parallel computing algorithm libraries**
- **Parallel visualization tools, parallel pre- and post-processing**

Let’s Get Started…

- **Objectives of parallel computing:**
  - Ultimately, in parallel computing, we intend to achieve:
    - Faster execution speed
      - Enable multiple analyses in a fixed amount of time
      - Decrease time necessary to complete one solution
      - Increase the level of modeling of our physical system
  - Larger problem size
    - Enable higher grid resolution than possible in single processor machines
    - Introduce additional physical models that were impossible to tackle before
  - Numerical experiments
    - Simulate phenomena that cannot be recreated or measured in the laboratory

Lecture 1 – Objectives of Parallel Computing

- Lower cost
  - Strictly speaking, it is nearly impossible to obtain lower cost when using a parallel computer (parallel processing overhead, additional expense of interconnection network, etc.)
  - Lower cost can be derived from additional benefits that result from the ability to execute a given program in a shorter amount of time
  - However, we must strive to maintain the cost of parallel computing from departing severely from single processor computations
  - Economies of scale?

Objectives of Parallel Computing

- **What parallel computing is NOT**
  - A brute force method to overcome limitations of your baseline algorithms/solution procedure
  - A “cool” way of getting solutions faster
  - An absolute need for every existing application

Objectives of Parallel Computing

- **Our objective in this course is to:**
  - Introduce students to the basic tool-set used in parallel computing for engineering problems
  - Determine what engineering problems can take advantage of parallel computing
  - How to set up an engineering problem to compute in parallel
  - How to program a computer code to solve an engineering problem with parallel computers

Current Industry Practice

- **Parallel computing is used extensively in certain industry:**
  - Aerospace:
    - Boeing, General Electric, Pratt & Whitney
    - Computational fluid dynamic simulations
      - External (fuselage, wing, nacelle, etc.)
      - Internal (engine compressor, turbine, combustor, inlet, nozzles)
    - Design optimization
  - Automobile:
    - Ford, General Motors
    - Computational fluid dynamic simulations
      - External (body drag)
      - Internal (underhood, passenger compartment, fans)
  - Electronics:
    - HP, Silicon Graphics
      - Chip and board heat transfer

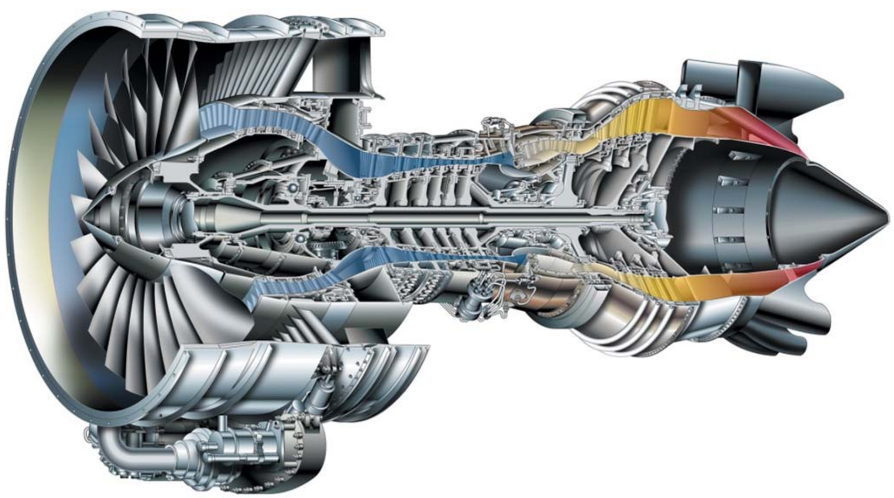
Examples

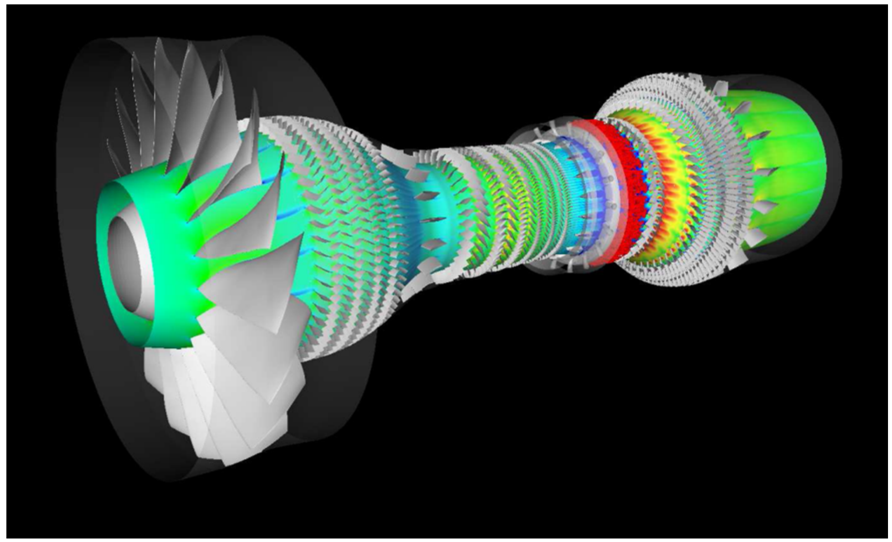
- **The examples that I will show in the following slides focus on parallel computing of fluid dynamic problems.**
- **However, there are many other sciences such as**
  - Heat transfer
  - Chemically reaction kinetics
  - Electromagnetics
  - Acoustics
  - Structural dynamics

  where parallel computing is used just as much or even more!

High-End Examples of Parallel Computing
Entire Jet-Engine Main Flow-Path

- **DoE Accelerated Computing Initiative:**
  - Develop the parallel computer hardware and software technology to solve large-scale, multi-disciplinary scientific problems never-before attempted
  - Example: 3-D flow through an entire jet engine

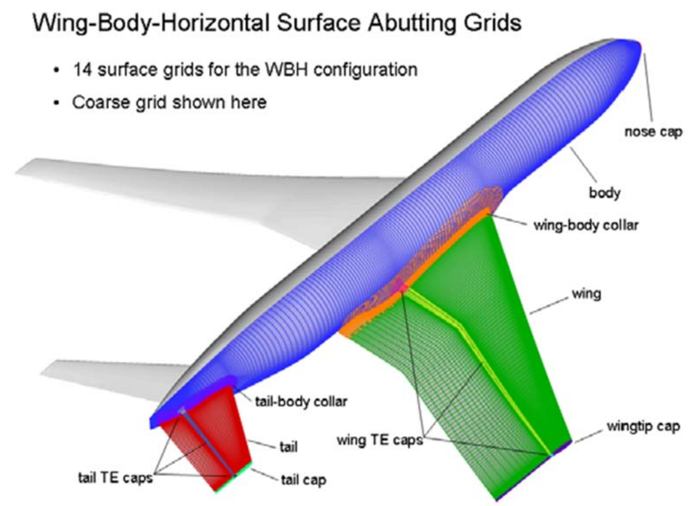


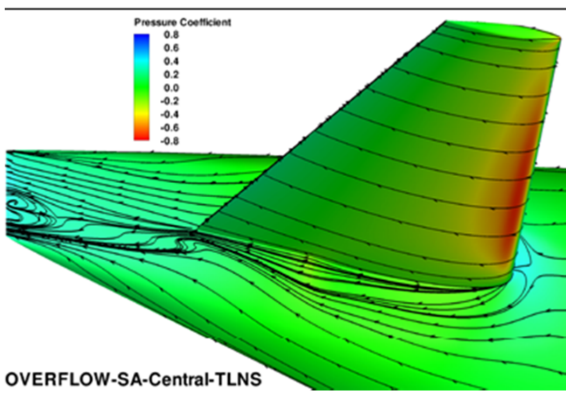


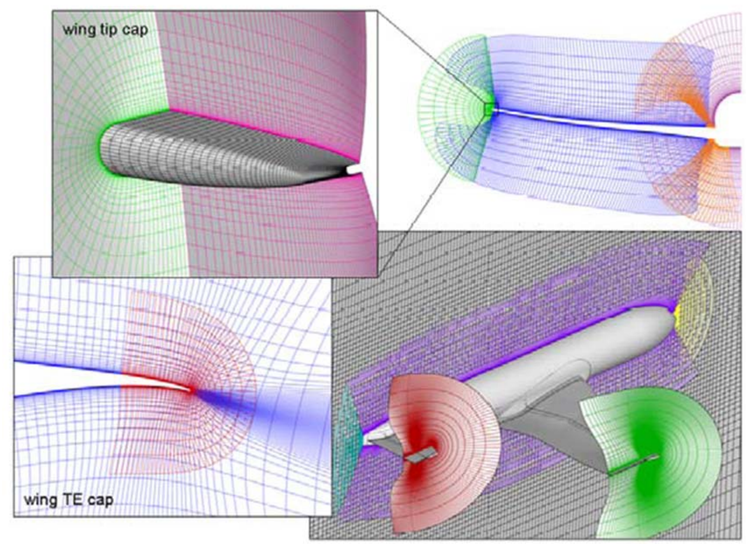
- **20^0 Sector Unsteady-Flow Simulation**
- **14 M Points (coarse-grid)**
- **75M Points (fine-grid)**
- **700 Processors (coarse-grid)**
- **4,000 Processors (fine-grid)**
- **~14 days Turn-around**

Entire Aircraft

- **AIAA Drag Prediction Workshop (Vassberg et al. at Boeing, NASA, and universities)**
  - Prediction of C_L and C_D of complete aircraft
  - Accuracy issues due to
    - Grid quality and topology
    - Solver





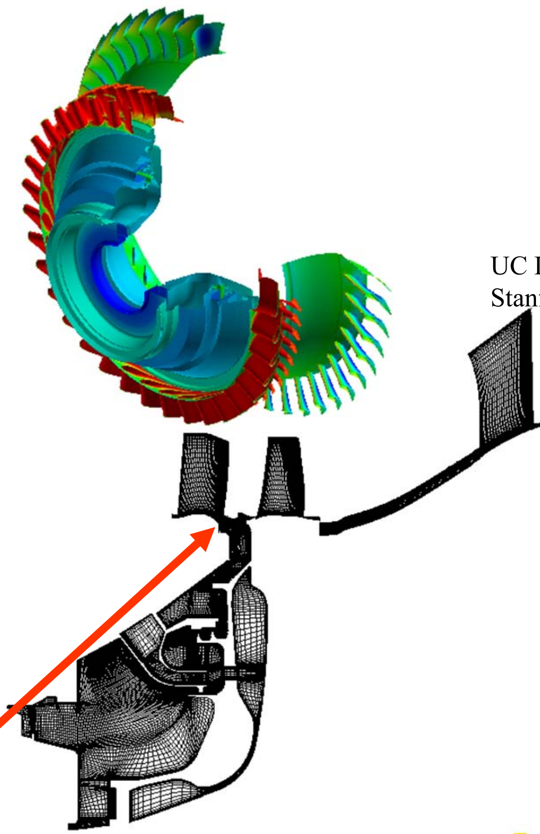


- **7 M Points (coarse-grid)**
- **2.4B Points (ultra-fine-grid)**
- **16 Processors (coarse-grid)**
- **4140 Processors (fine-grid)**

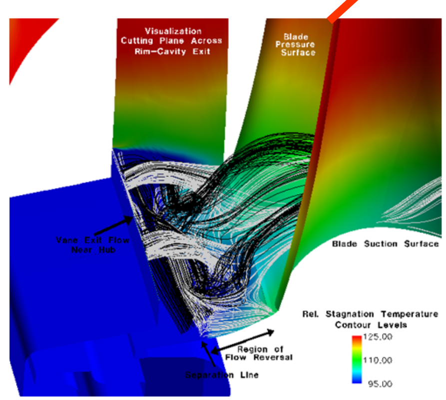
Boeing, NASA

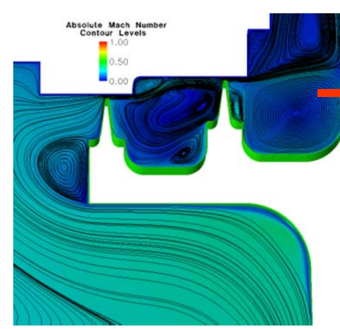
Main/Secondary Turbine Flow-Path Simulation

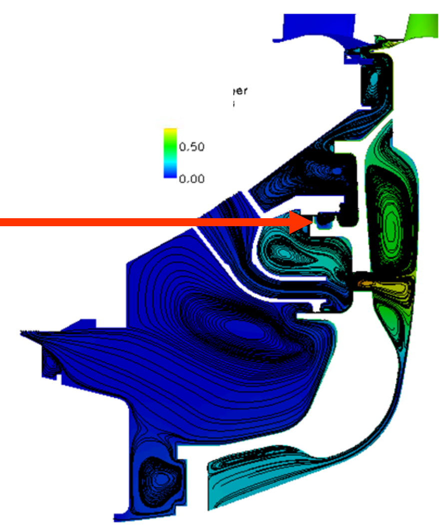
- **Integrated High-Pressure Turbine Main and Secondary-Air System Flow Paths**
- **Investigation Featured:**
  - Description of main/secondary-air interaction and a source of hot-gas ingestion into disk cavity
  - Flow physics in/around seals of secondary-air system
  - Weaknesses with traditional flow leakage modeling of secondary-air into main flow path



UC Davis and
Stanford University



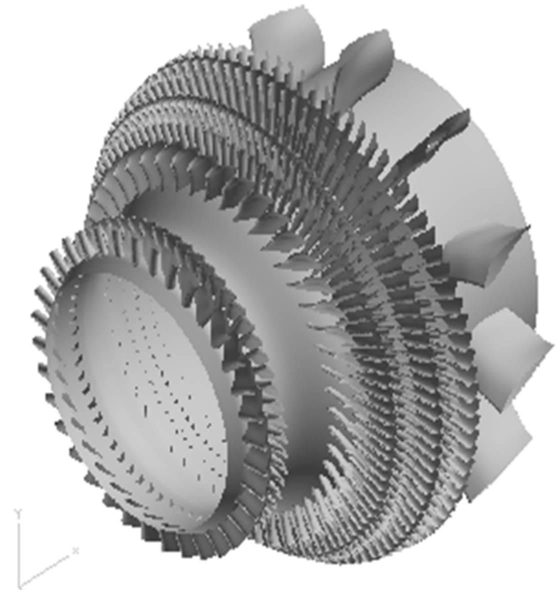


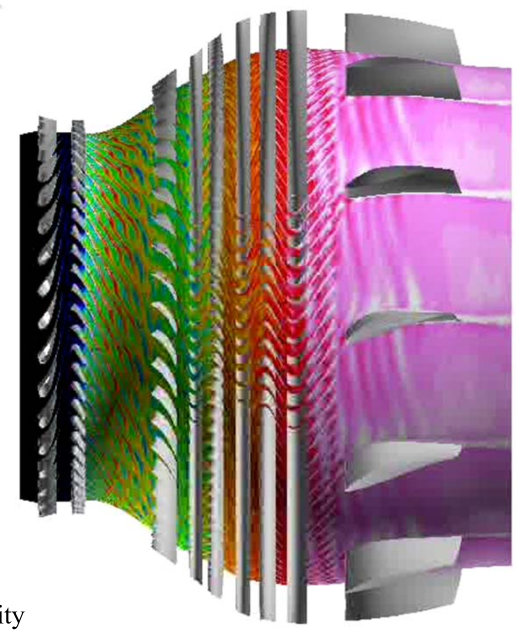


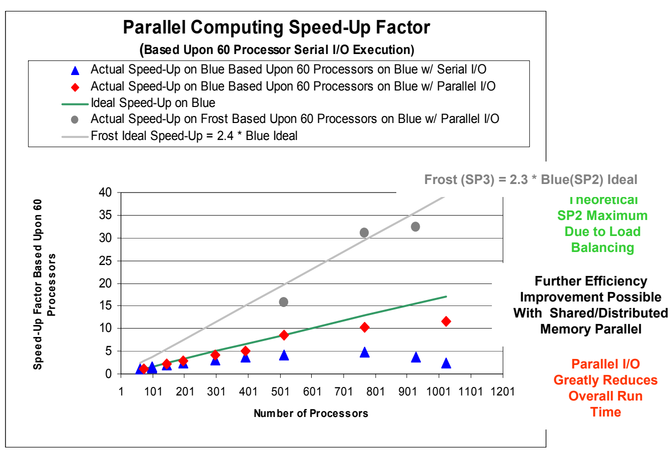
- **Steady-Flow Simulation**
- **9.4 M Points**
- **238 Blocks**
- **144 Processors, IBM SP3**
- **~100 days Turn-around**

Unsteady-Flow Simulation
Integrated High- and Low-Pressure Turbine

- **1/6 Circumference Modeled**
- **93.8 M Points**
- **2192 Blocks**
- **384-640 Processors**
- **5,700 Time-Steps (w/ 30 inner iterations per time-step) Required**
- **~85 days (clock time), ~1.3M Hours (cpu time) on Frost (IBM SP3)**
  - 640 processors (40 nodes)

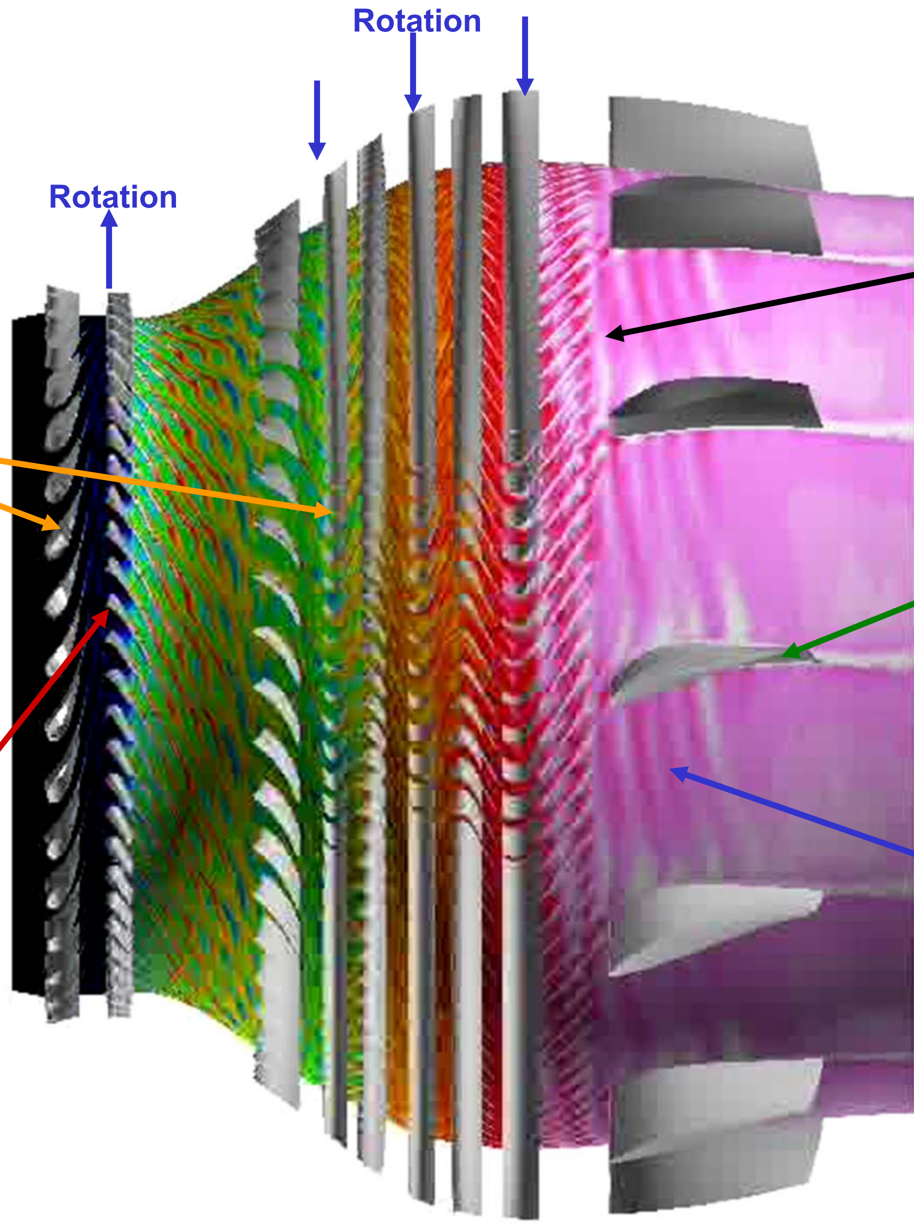






UC Davis and
Stanford University

Flow Features Highlighted by Simulation



Wake-Blade Interaction Source of Resonance Stress and High-Cycle Fatigue

Unsteady Shock Boundary-Layer Interaction Source of Aero Performance Degradation and High-Cycle Fatigue

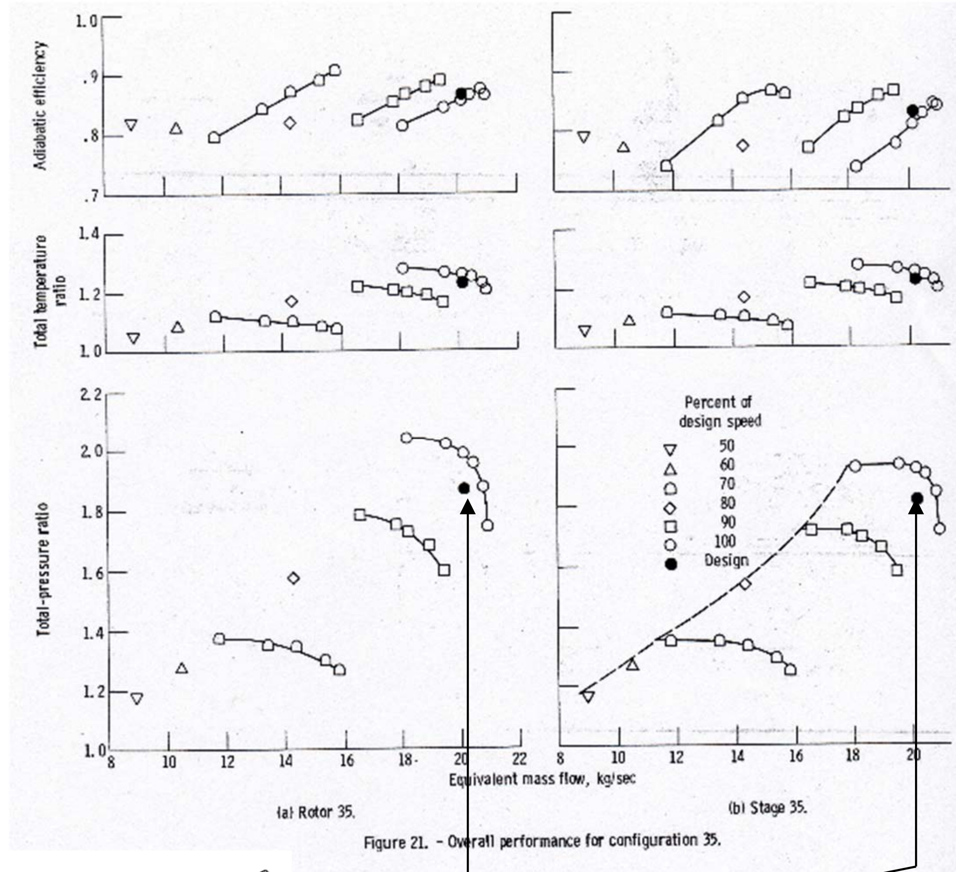
Unsteady Flow Propagates into Downstream Mixer and Interacts with External Flow to Create Jet-Noise Source

For Military Engines, Unsteady Flow Can Lead to Augmentor (Afterburner) Instability

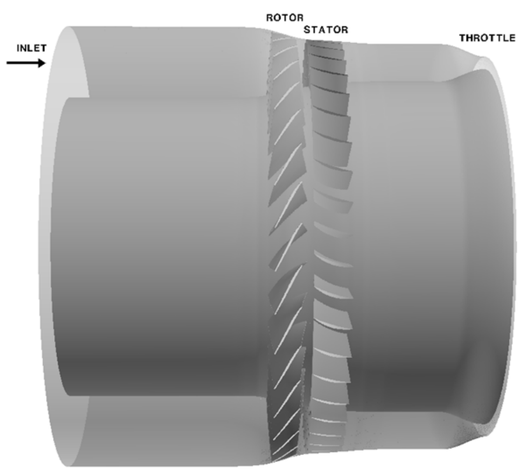
Entropic Waves Act as Noise Source

Stage 35 Stall-Inception/Control Investigation

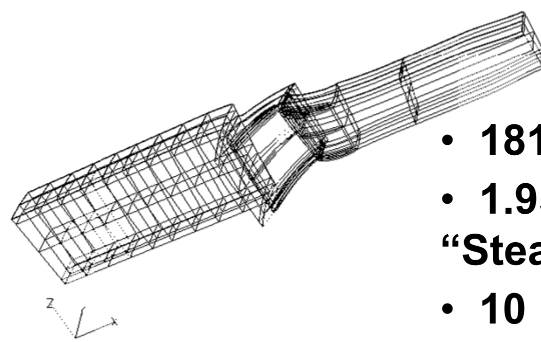
- **NASA Stage-35 Compressor**
  - Picked by GE and NASA as meta-goal target problem
    - Flow physics leading to stall
    - Flow control of stall
  - Geometry and reports acquired from NASA
  - Single-Stage
    - 36 rotors, 46 stators



Current Investigation

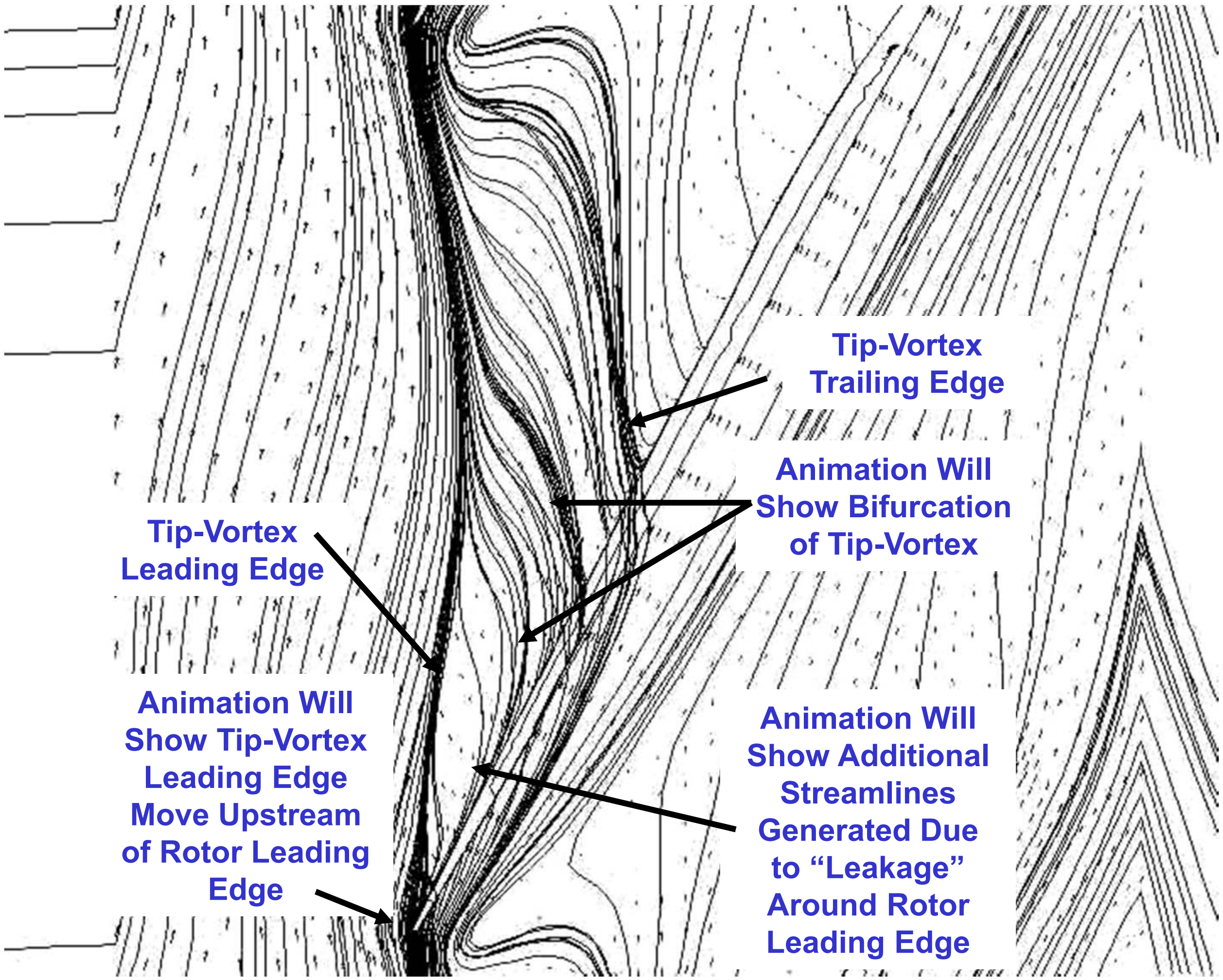


Computational Model



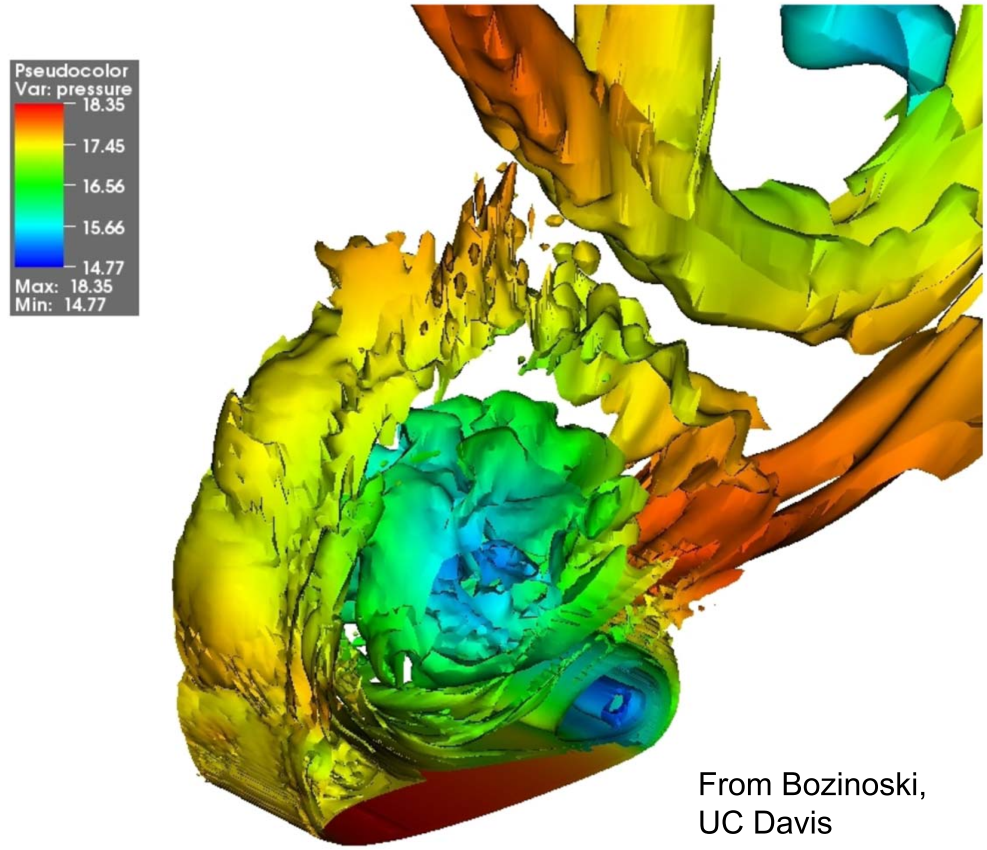
- **181 Grid Blocks**
- **1.95M Grid Points for "Steady" Simulation**
- **10 processors**

Stall Inception

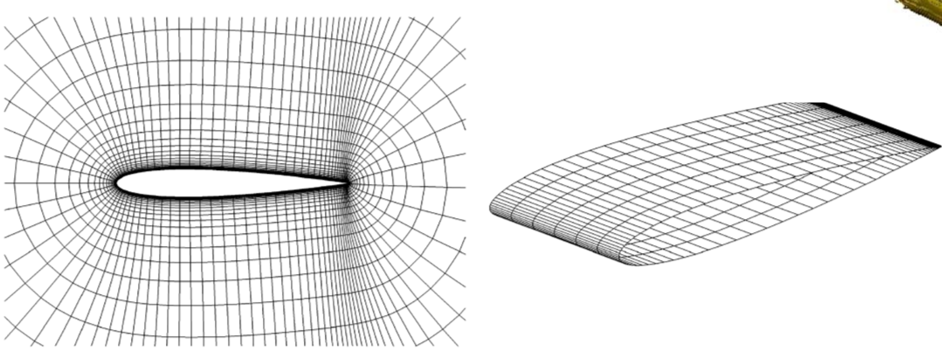


Airfoil Stall

- **Detached-Eddy Simulation of NACA0012 at high angle of attack**
  - 1.8M grid points
  - 20 processors on wopr/vortex



From Bozinoski, UC Davis



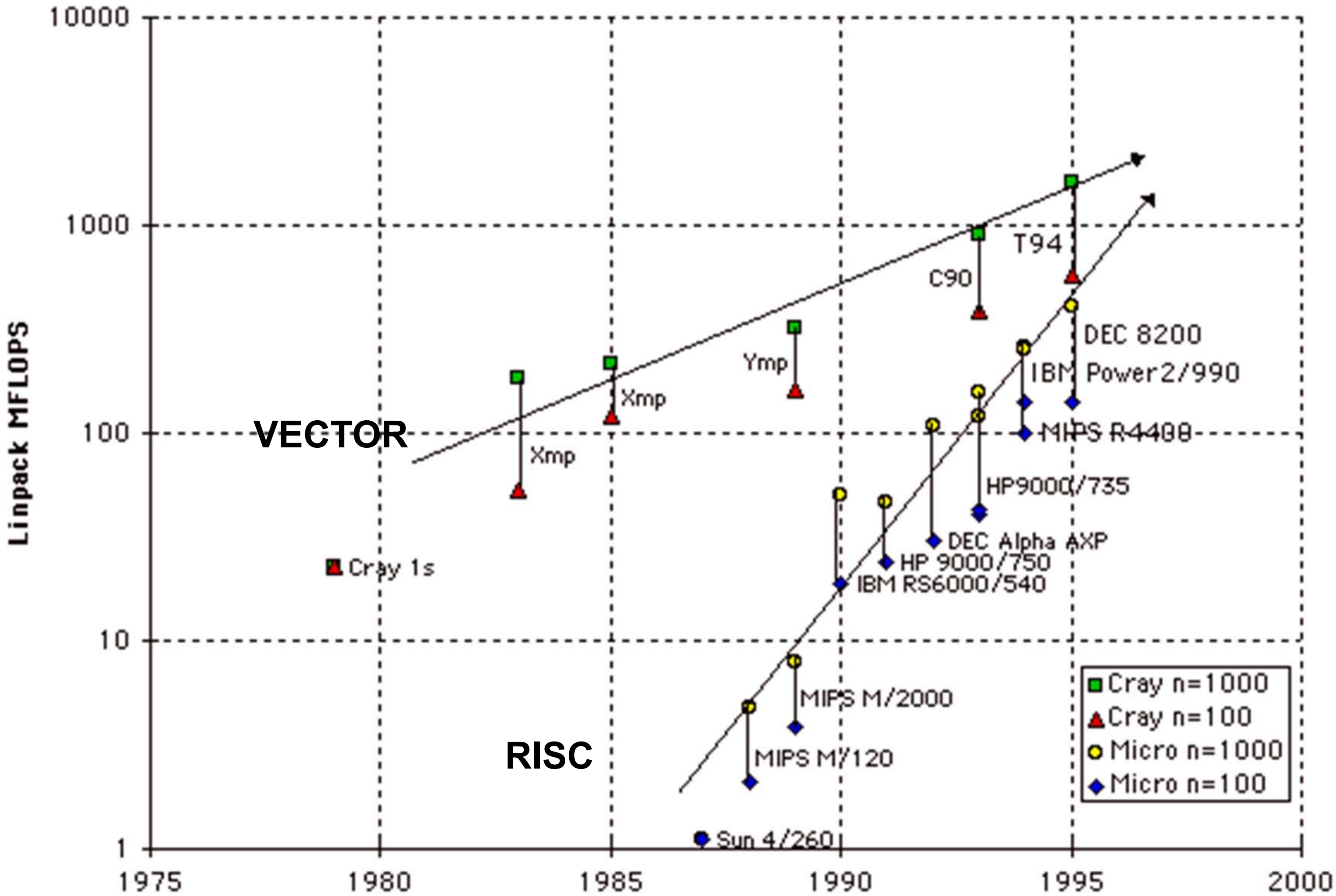
(a) Stream-wise view

(b) Span-wise view

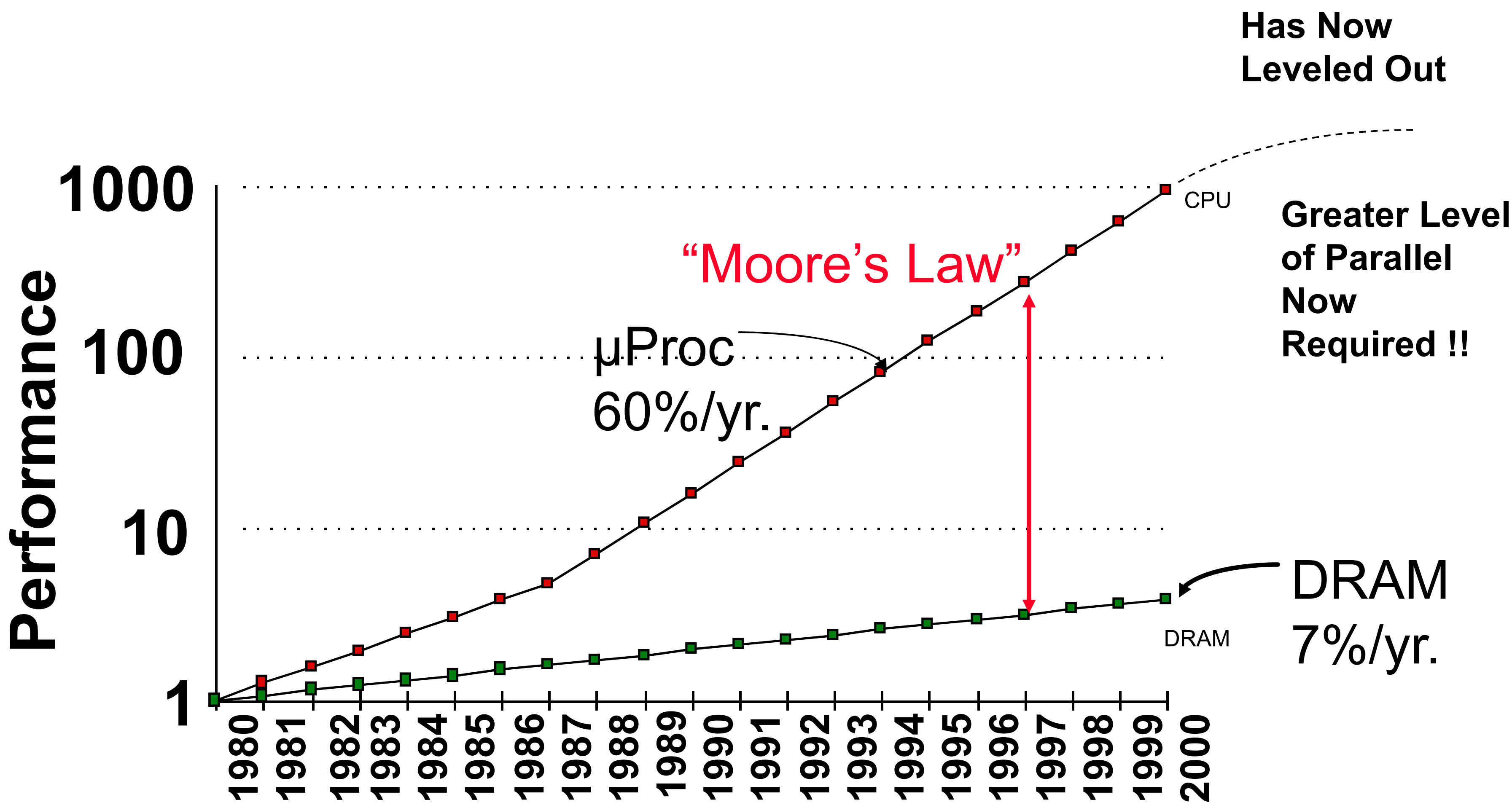
UC Davis

	C_L	C_D
2D URANS	1.32	2.24
2D DES	1.34	2.28
3D URANS	1.40	2.37
3D DES	1.08	1.87
Experiment	0.90	1.60

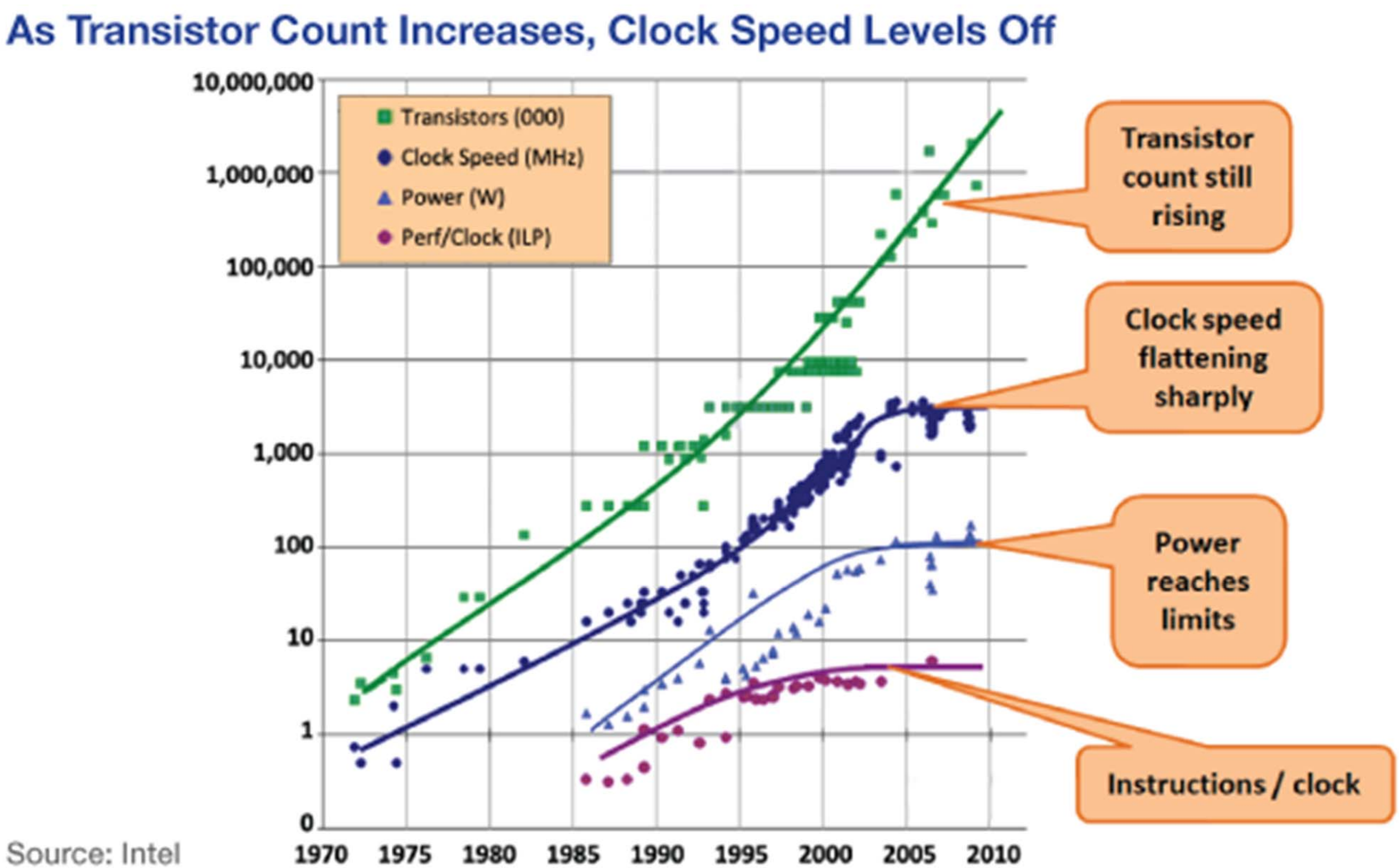
Early Microprocessor Performance



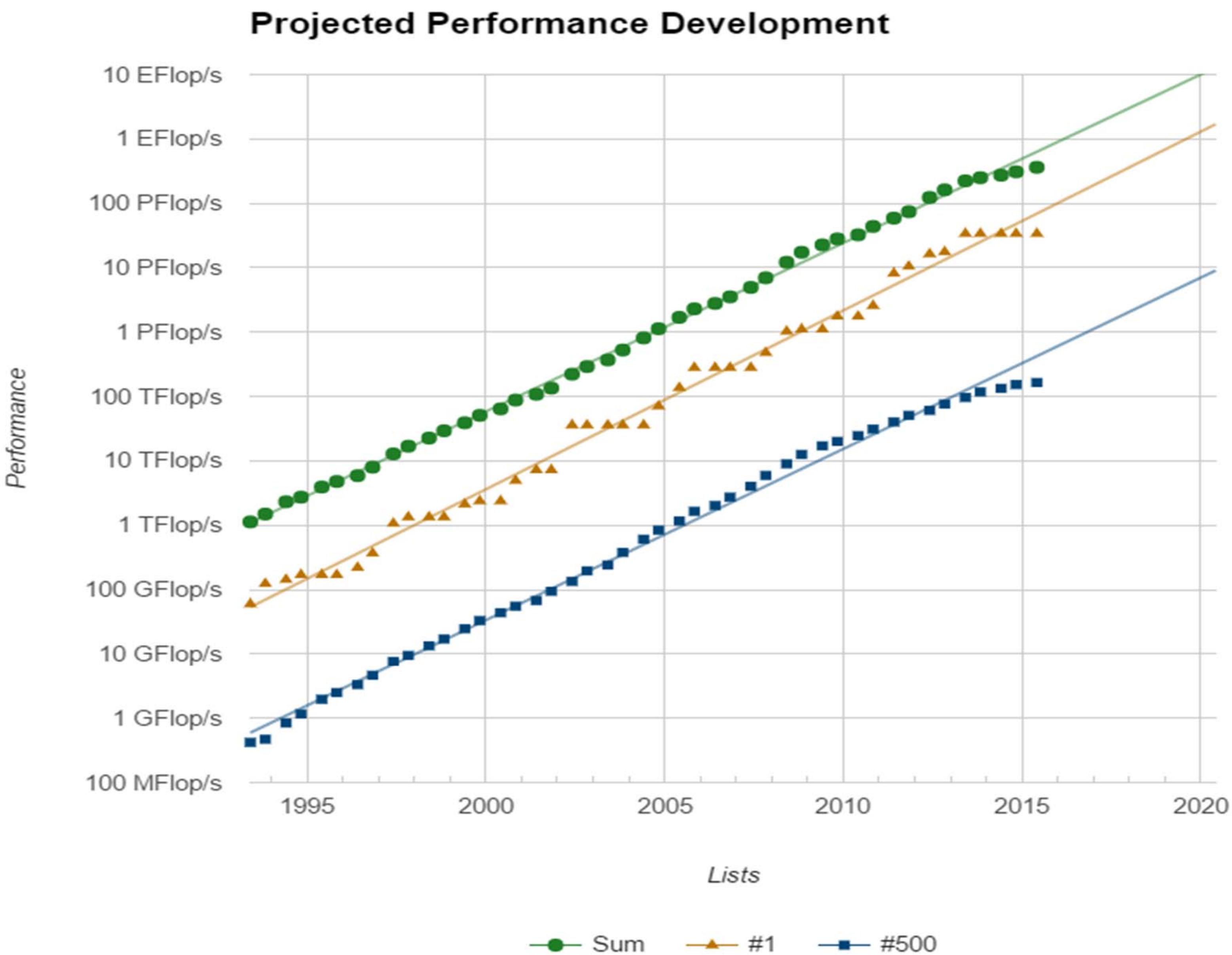
Early Memory Subsystems



Microprocessor Speed



Computer Performance History



Computing Resources

- **Vortex cluster (going off-line)**
  - 210 processors (vortex)
  - Linux operating system
  - Gigabit Ethernet communication
  - Graduate student research
  - We will be using the **hpc1** system (similar to vortex) in this class. You should have accounts on this system by now.

- **Davistron desktop-cluster (defunct)**
  - Built at UCD by students
  - 8 second-generation processors
  - Linux operating system
  - Gigabit Ethernet communication
  - System lasted approximately 5 years before becoming obsolete

Davistron Original Hardware

4x	Barebones Computer with k9VGM-V Motherboard	$291.96 ($72.99 each)
1x	Samsung DVD Drive (SH-S203B)	$29.99
4x	AMD Athlon 64 X2 4000+ Brisbane 2.1 Ghz Processor (Dual Core)	$279.96 ($69.99 each)
4x	2Gb Corsair Value Select 240-Pin DDR2 Memory (PC2 5300)	$199.96 ($49.99 each)
4x	80 Gb Western Digital Caviar 7200 RPM Sata 3.0 Gb/s Hard Drive	$171.96 ($42.99 each)
5x	TRENDnet FastE Ethernet Card	$29.95 ($5.99 each)
Total		**$1,003.78**

Logging into hpc1

- **You will need to obtain a login account that uses a secure shell protocol. I've sent out information on this already.**

- **You will need to obtain secure shell (ssh) or putty in order to log into any of our parallel systems**
  - This software uses encryption to protect the systems and users from malicious attacks
  - Each of you should be able to log into hpc1 under your own account via the secure-shell utility (an encrypted version of telnet)

    ssh hpc1

- **You will be located in your "home" directory when you first enter hpc1,**

  /home/"your username" e.g. /home/davisrl

Common Linux Commands

- **cd** :: **Go to home directory**
- **cd ./...** :: **Go to directory "..." in current folder**
- **cd ..** :: **Go up one directory from current folder**
- **cd /...** :: **Change to absolute directory "..."**
- **ls** :: **List contents of current directory**
- **ls -lh** :: **List file permissions and file size**
- **du -h** :: **List size of current directory contents**
- **df -h** :: **List available space on all installed drives**

- **Other basic Linux commands are located at smartsite under the "Additional Material" folder**
  - BasicLinuxCommands.pdf

Creating Directories, Programs, Files

- **Once you have successfully logged into hpc1, you should create your own directory in which you will create computer programs and run**

Create directory:	mkdir “codename”	where codename is something like hw1
Enter directory:	cd “codename”	set directory to hw1

- **Programs/files can be created using one of many editors. Examples include “vi” or “emacs”. There are advantages/disadvantages to each.**

vi mfp.f

emacs mfp.f

Compiling a Fortran Program

- **Each computer vendor typically has its version of a Fortran compiler that converts the Fortran into machine language**
  - Examples include:
    - IBM – xlf90
    - SGI – f90
    - Linux – pgf90, gfortran

- **Programming, compiling, and linking a program is usually done on the “front-end” computer, i.e. hpc1**

Compiling a Fortran Program

- **We have 3 compilers on hpc1**
  - Portland Group, pgf90, pgcc, etc. (commercially available)
  - GNU, gfortran (open source)
  - Intel, ifort, icc, etc.
- **These compilers are located at**
  - **Serial Execution:**
    - pgf90: /opt/pgi/linux86-64/2015/bin
      - For pgf90, pgf95, pgcc, pgCC basic compiling (without MPI bindings)
    - GNU without MPI bindings: /usr/bin
      - For gfortran, gcc, etc.
    - Intel without MPI bindings: /share/apps/intel-2015/bin
      - For ifort, icc, etc.
  - **Parallel Execution:**
    - Openmpi using PG, or Gfortran with MPI bindings: /usr/bin/mpif90
      - For mpif90, mpicc, etc.
      - This can also be used for Serial Execution

← Recommended for both serial and parallel

Compiling a Fortran Program

- **Once you have created the serial program, you may compile it on hpc1 using mpif90**
- **This will create an object, mfp.o If a program consists of several objects (subroutines) along with the main program, they can be "linked" together**
  - mpif90 –o mfp mfp.o, sub1.o, sub2.o …
- **Serial jobs can be run on the front-end (hpc1). However, parallel jobs must be run in batch mode in the background.**
- **A serial executable "mfp" can be run by just entering**

  ./mfp

Using MAKEFILES

- **Compiling and Linking to create executables can be all performed in a script called a "makefile"**

- **An example makefile for "codename" has been put on smartsite under the "codes" folder**

- **If your makefile is simply named "makefile", then you can run it by typing "make"**

- **If your makefile is named something else (eg makefile_code) then you can run it by typing "make –f makefile_code"**

SLURM Job Scheduling

- SLURM is a job scheduling tool that allows multiple users to submit batch jobs without having to worry about running on top of each other

- Jobs are prioritized based on the number of CPU's requested and the amount of time the job has been in the queue

- Help with hpc1 can be obtained from

help@cse.ucdavis.edu

or from my graduate student, Daryl Lee (dywlee@ucdavis.edu)

Submitting Batch Serial Jobs on hpc1

- **Batch script to submit a serial job to compute node on hpc1:**

```
#!/bin/bash -l
#SBATCH -J r_adia
#SBATCH -o r_adia-%J.out
#SBATCH -e r_adia-%J.err
#SBATCH -n 1

echo "starting at `date` on `hostname`"

#Run Command (assumes pgi and openmpi)

module load pgi openmpi hwloc          ← Using openmpi with PG libraries
./E> "listing“                         ← where E is the name of executable
                                       ← where listing is any name with output
```

Submitting Batch Parallel Jobs on hpc1

- **Batch script to submit a parallel job to multiple cores on hpc1:**

```
#!/bin/bash -l
#SBATCH -J r_adia
#SBATCH -o r_adia-%J.out
#SBATCH -e r_adia-%J.err
#SBATCH -n N                                   ← where N is the number of cores

echo "starting at `date` on `hostname`"

#Run Command (assumes pgi and openmpi)

module load pgi openmpi hwloc                  ← Using openmpi with PG libraries
mpirun -n  N ./E> "listing“                    ← where E is the name of executable
                                               ← where listing is any name with output
```

Serial makefile explained 1/2

```
#Sample Makefile for Bounce
#Written by Michael Ahlmann and modified by Roger Davis
#Written for MAE267

#Assumptions
#               1) Prorgram is written in fortran
#               2) All files are in current directory
#               3) All files have .f, .f90, or .f95 extension

#Set Compiler Flags (to pgi version of openmpi)
FC = mpif90

#Set Optimization or Debug Flags
#    -fast = Full Optimizations
#    -g    = Debug Mode
OPTMZ = -fast

#Set Compiler Flags
#      -c      = Compile Only Don't Link (Required)
CFLAGS = -c

#Set Compiler Libraries
LIBS =

#Set Program Name
PROG = bounce_pgi

#List Object Files
#               This section should include all files to be
#               compiled with the .f, .f90, or .f95 extension replaced with
#               a .o extension. Use \ at the end of each line to
#               extend across multiple lines
OBJS = bounce.o
```

Use absolute path names to compilers when possible.

mpif90 is a wrapper compiler that points to either the gnu or pgi fortran compiler and links to the mpi libraries. It is not necessary to use mpif90 for serial programs, but it does not hurt to do so.

Program name can be whatever you want. Just make sure it is consistent with the name in your runjob script.

Multiple object files can be listed at once.

Serial makefile explained 2/2

```
#List Modules (follow same rules as for objects)
MODULES =

#Command Telling Make to Compile Program
$(PROG): $(MODULES) $(OBJS)
          $(FC) $(OBJS) $(MODULES) $(OPTMZ) -o $(PROG)

#Command Telling Make to Compile Modules
$(MODULES): %.o: %.f
          $(FC) $(CFLAGS) $(OPTMZ) $< -o $@

#Command Telling Make to Compile Object Files
$(OBJS): %.o: %.f $(MODULES)
          $(FC) $(CFLAGS) $(OPTMZ) $< -o $@

#Command to Clean Directory if User Desires
clean:
          rm -f *.o
          rm -f *.mod
          rm $(PROG)
          rm bounce.o*
          rm bounce.po*
```

Module files can be listed here

This section should not need to be changed. If you are interested in the details of how these commands work, either look it up on google or e-mail questions to davisrl@ucdavis.edu

The clean command is not required, but it can be convenient to clean up files from compiling or running your code. Simply run *make clean* to invoke the code from this section.

Other Useful (Free) Tools

Plotting

ParaVIEW : Good for visualization of plot3d files.

Available from www.paraview.org

VisIT : Good for visualization of CGNS files. Can visualize plot3d files, but a quick script is required (look in plot3d example folder for a sample script). Available from www.llnl.gov/visit.

TechPLOT: Good for contour or line plots and animations.

Available from Jacob in MAE.

FIELDVIEW: Good for contour, line, and animations.

Available from Jacob in MAE.

Text Editing

Textpad : Good for viewing source code on a windows computer.

Available from www.textpad.com

Text Wrangler : Good for viewing source code on an Apple computer.

Available from www.barebones.com/products/TextWrangler/

Kate : Good for viewing source code on a linux computer (install via Yum)

SSH and File Transfer

Cyber Duck : File transfer on an Apple Available from http://cyberduck.ch/

WinSCP : File transfer on Windows.

Available from www.winscp.net

Homework 1

- **Read Chapters 1- 4 in Fortran 95/2003**

- **Try out your accounts on hpc1**
  - Try logging on using Secure Shell (ssh at www.ssh.org) or Putty (www.putty.org)
  - Learn various Linux/Unix commands to navigate, etc.
    - See BasicLinuxCommands.pdf in Additional Material/Linux Commands folder on smartsite
  - Set up directories for your homework and projects

- **Problems**
  - Problems Below Due Thursday, Oct. 1
    - You can develop and execute these on the hpc1 front-end computer or on the cluster nodes by using the serial job submit procedure

Problem 1

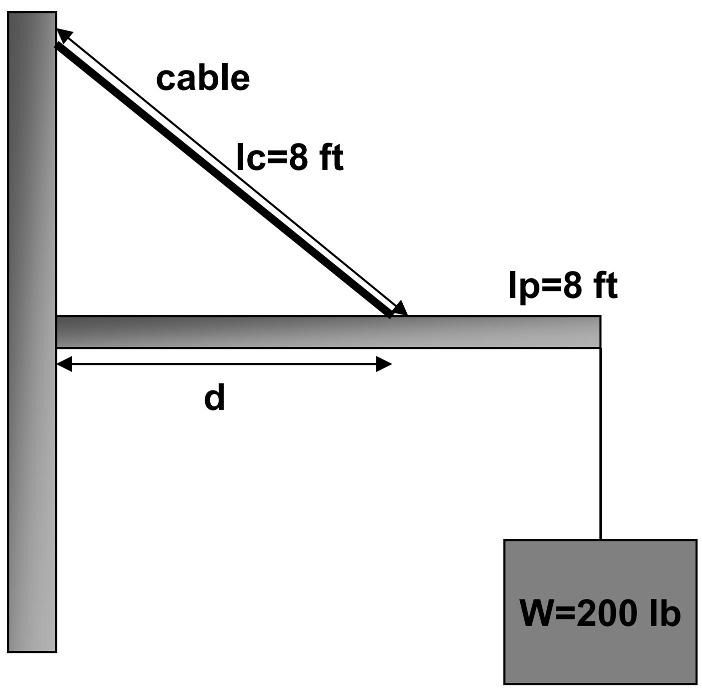
- **Period of a Pendulum**
  - The period of an oscillating pendulum T (in seconds) is given by the equation

$$T = 2\pi\sqrt{\frac{L}{g}}$$

where L is the length of the pendulum in meters and g is the acceleration due to gravity in meters per second squared. Write a Fortran program to calculate the period of a pendulum of length L. The user will specify the length of the pendulum when the program is run. Use good programming practices in your program. (The acceleration due to gravity at the Earth's surface is 9.81 m/sec^2.)

Problem 2

- **Tension on a Cable**
  - A 200 pound object is to be hung from the end of a rigid 8-foot horizontal pole of negligible weight, as shown below:



Problem 2 (cont)

The pole is attached to a wall by a pivot and is supported by an 8-foot cable that is attached to the wall at a higher point. The tension on this cable is given by the equation:

$$T = \frac{W(lc)(lp)}{d\sqrt{lp^2 - d^2}}$$

where T is the tension on the cable, W is the weight of the object, lc is the length of the cable, lp is the length of the pole, and d is the distance along the pole at which the cable is attached. Write a program to determine the distance d at which to attach the cable to the pole in order to minimize the tension on the cable. The program should calculate the tension on the cable at 0.1 foot intervals from d=1 foot to d=7 feet and should locate the position d that produces the minimum tension.