High-Performance Computing

Lecture 1: Introduction

Me

- Thomas Fogal
  - “Tom”, please
  - thomas.fogal@uni-due.de

Others

- Rainer Schlönvoigt
  - rainer.schloeni@gmail.com
- Prof. Dr. Jens Krüger
  - jens.krueger@uni-due.de

I don't know you.

- Picture on moodle?
- Matrikelnummer on moodle?

HiWis? Masters / Research?

- Our groups' focus
- Ace the class.

Course Organization

- 1 part N-Body, 1 part Final Project
  - Grading weights? 70/30 30/70?
- Clusters, MPI, OpenMP
  - CUDA/OpenACC/OpenCL off-topic
- Language choice: C, C++, Fortran 90+. GNU.
  - C intro?
- Groups
  - 1 or 2

Course Goals

- Common language
- Software engineering, practicals
- HPC: theory && practice
  - Distributed memory systems, MPI
  - Memory wall. NUMA
  - Shared memory, threading, OpenMP
  - Filesystems, I/O
  - Load balancing
  - Profiling and scalability
- 'Research' / final project

“Plan”

1. Intro, n-body, Linux essentials
2. Distributed memory
3. Particle Vis?
4. Distributed Filesystems
5. MPI File I/O
6. Shared memory
7. Memory access, t/s consistency
8. Scalability, profiling
9. Useful/general parallel algorithms
10. Future clusters

Assignments

- 5 or 6 total
  - Couple of weeks each
  - Build on each other
    - Live with your code!
- Groups
  - 2 or 1 students
  - Tell me before next class
- No sharing code!

Grading / Concerns

- Homework, grading issues: talk to Rainer
- Course issues: talk to me
- Escalate:
  - Rainer, myself, Jens, DUE administration

Assumptions

- Know an imperative language
  - Read C
- Do your homework
- ***Ask Questions***

Practicals / Recommendations

- Use C.
- Code locally, test on Cray
  - Don't waste CPU hours.
  - VM if you need it

Simulation Overview

Example Simulation Scenarios

- Molecular dynamics
- MHD
- Stress simulation (safety verification)
- Fluid flow
- Weather forecasting

Simulation Cycle

1. idea/theory/model
2. discretize domain
3. encode math into calculation
4. run simulation
5. verify result / explore data
    1. See our SciVis course :-)
6. GOTO 1

Parallel Simulation

- Reduce time to solution
- More nodes → more memory

Supercomputing

- Vector machines
  - Modern vector: SSE, Altivec
- beowulf

Parallelization

- Hard.
  - Race conditions
  - Coordination
  - Performance!
- How?
  - Automatic parallelization?
  - Threads?
  - MPI
    - System assigns procIDs → processors!

Threads

- Task-based parallelism
- For data parallelism?

Message Passing Interface

- Independent processes, different data
  - SPMD
- Each process has assigned ID
- Explicit synchronization
- Explicit memory transfer

Output

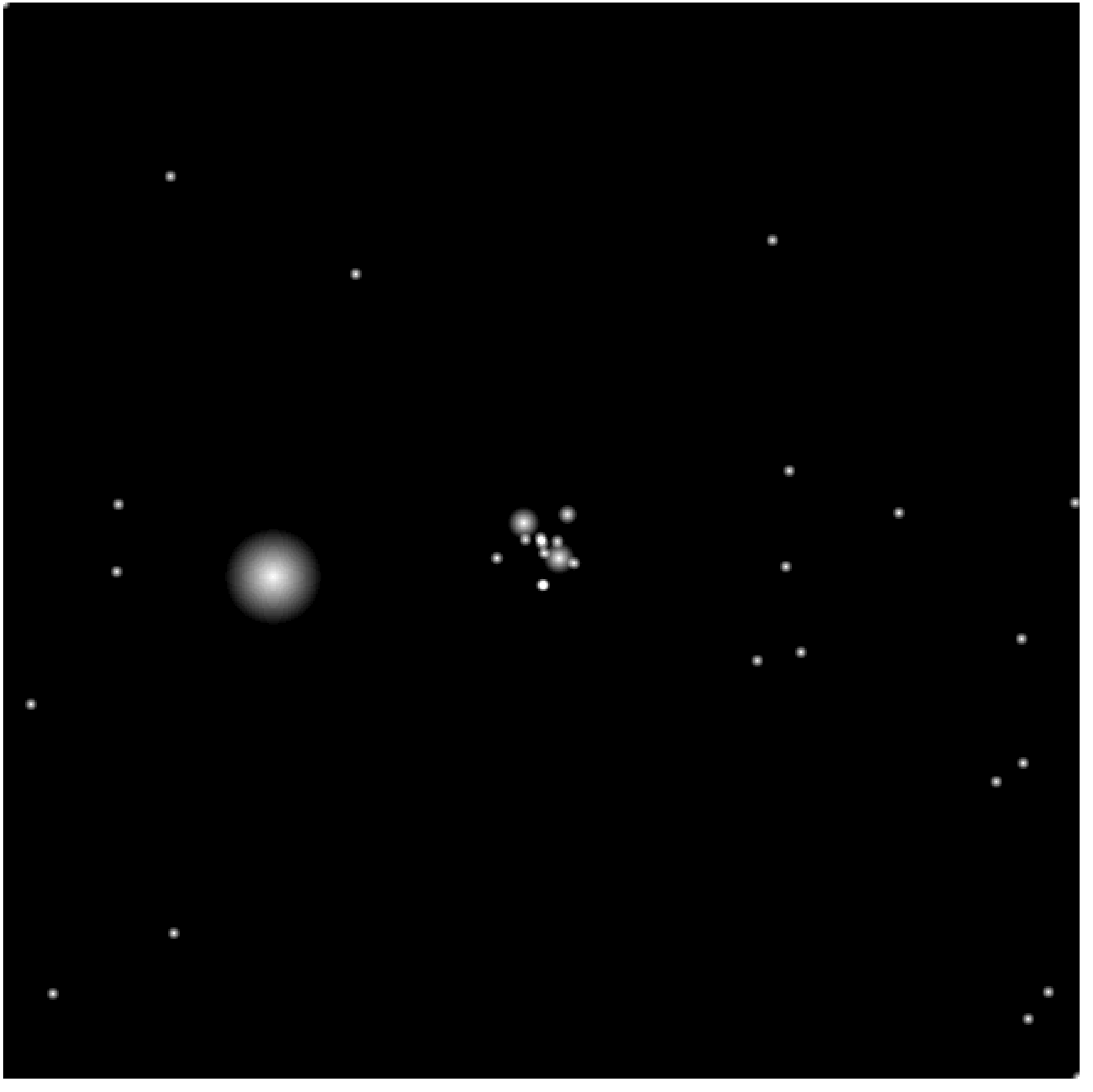
- Distributed file systems
  - GFS, GoogleFS (GFS..), Hadoop FS, Lustre, (NFS?)
- Usage patterns
  - Dump memory to disk (checkpoint)
  - Data arrays
  - Appends (log files)

Input

- Disk → memory (restart)
- Configuration
  - Derived from visualizing the data :-)
- Analysis / statistics

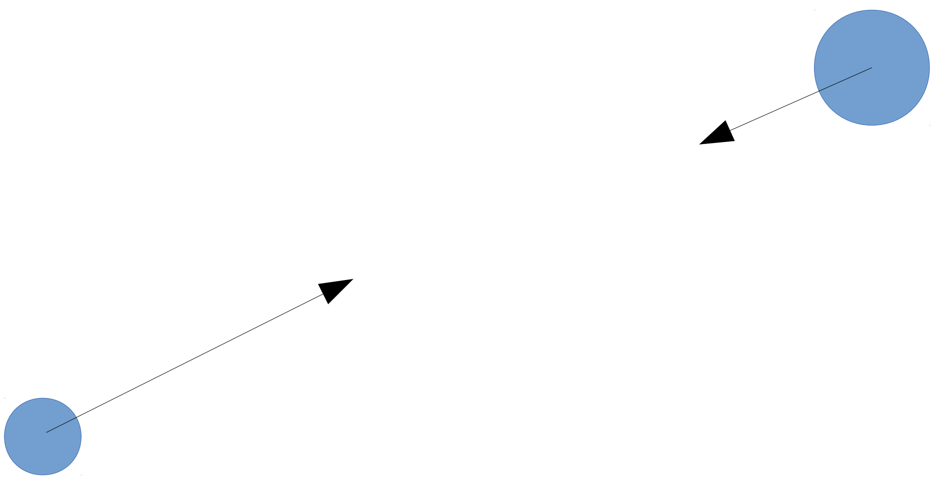

STAND BACK
I'M GOING TO TRY
SCIENCE

N-Body Problem

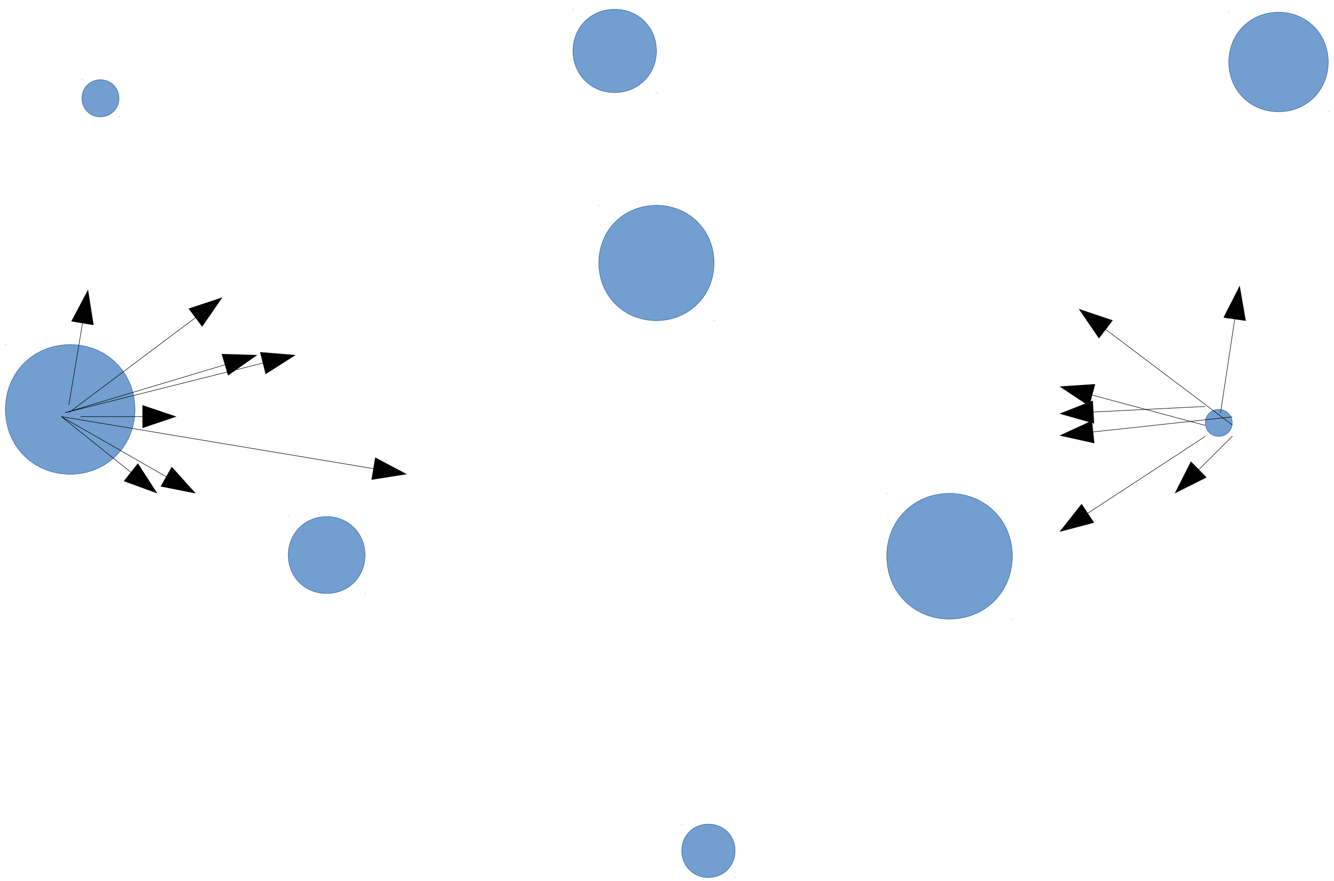


Newtonian Gravity

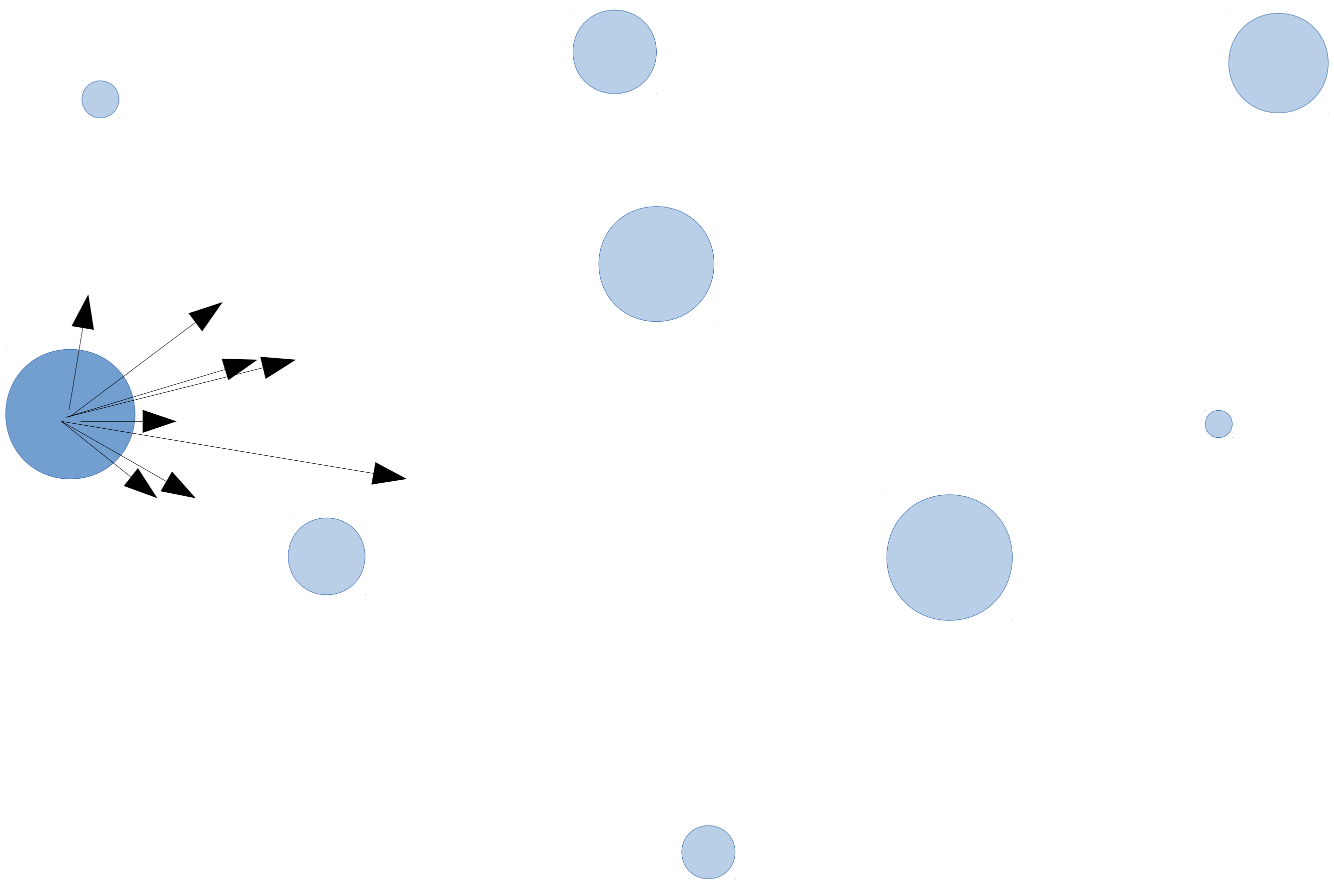
- $F_1 = F_2 = G (m_1 m_2)/r^2$



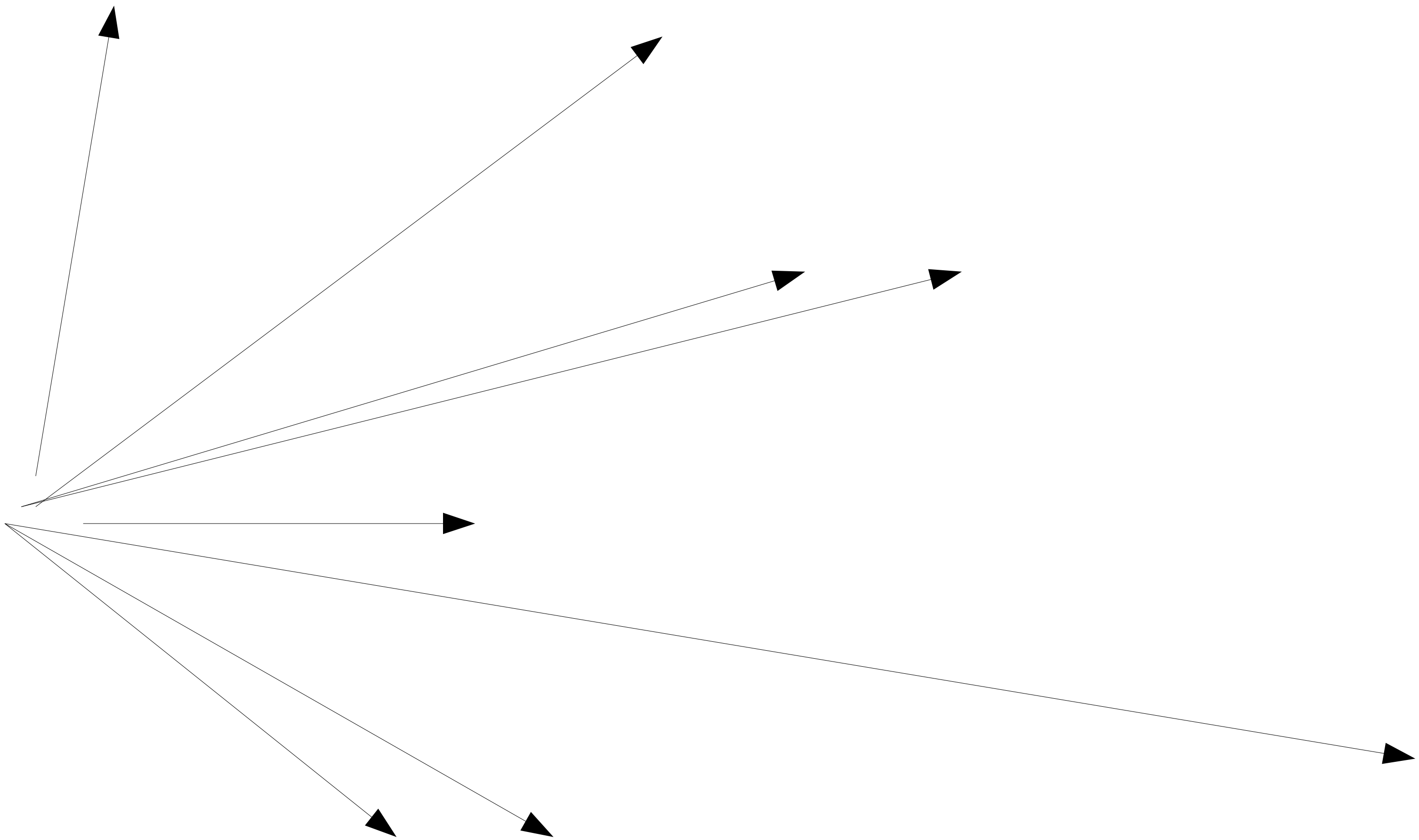
Many Particles



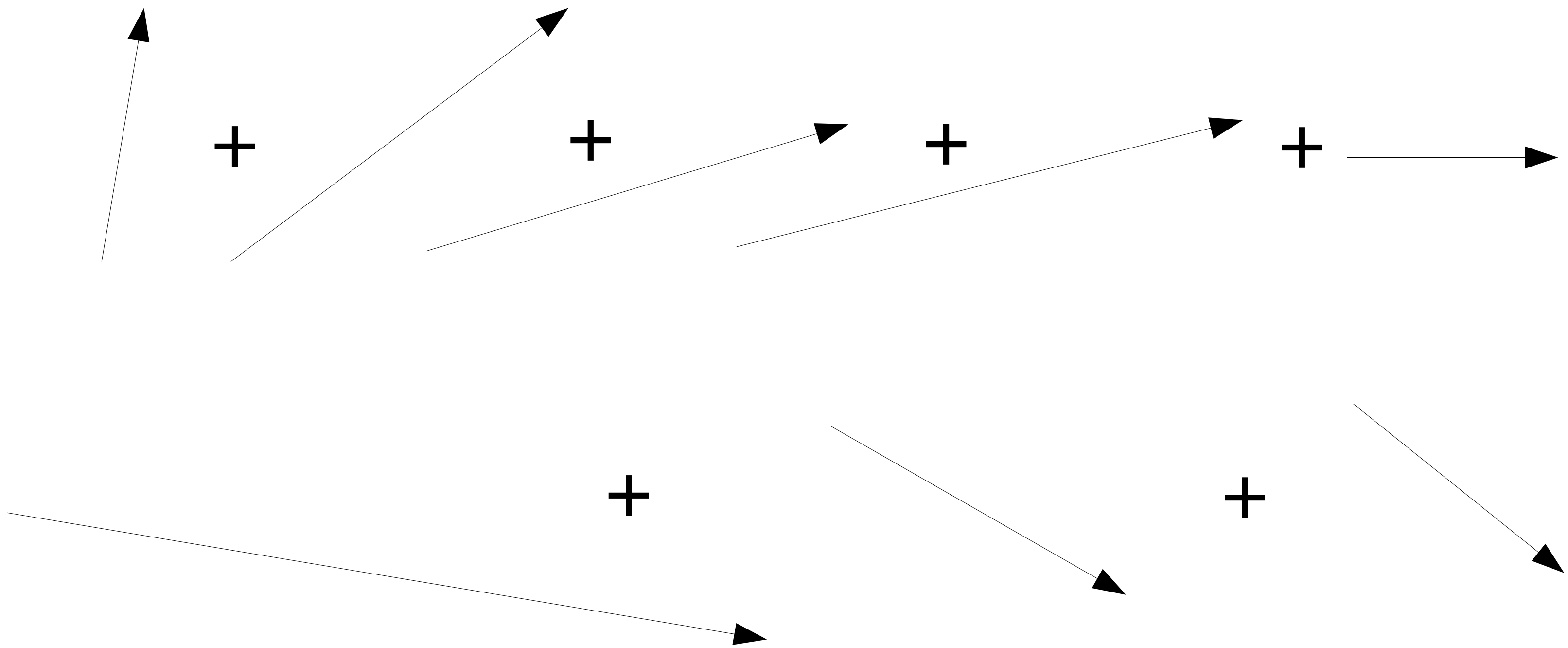
Summation of Forces



Vector Addition



Vector Addition



= F

Particle Force Summation

$$m_i p_i = G \sum_{j=1}^{N-1} \left(m_j m_i \left(p_j - p_i \right) \right) / \left| p_j - p_i \right|^3$$

$$G = 6.67 \times 10^{-11} N \left(\frac{m}{kg} \right)^2$$

Practicals

- How large is T?

Linux Essentials

Terminal

$ _

$ cmd1

$ cmd2

Output

```
$ cmd
cmd's output
more output
$
```

Canceling Commands

```
$ ./a.out^C
$
```

Navigation

```
$ cd directory
$ cd ..
$ ls
$ pwd
```

Compiling

$ gcc -Wall -Werror -ggdb3 file.c

Wall: turn on all warnings

Werror: warnings *are* errors

ggdb3: debug symbols on

O3: heavy optimization

Debugging

```
$ gdb -q ./a.out
(gdb) run
…
^C
(gdb) bt
(gdb) list
```

Valgrind

$ valgrind --leak-check=full
--track-origins=yes
--leak-resolution=high
--show-reachable=yes ./a.out ...