Intro to Some Advanced Topics

15-213 / 18-213: Introduction to Computer Systems
27th Lecture, Dec. 4, 2012

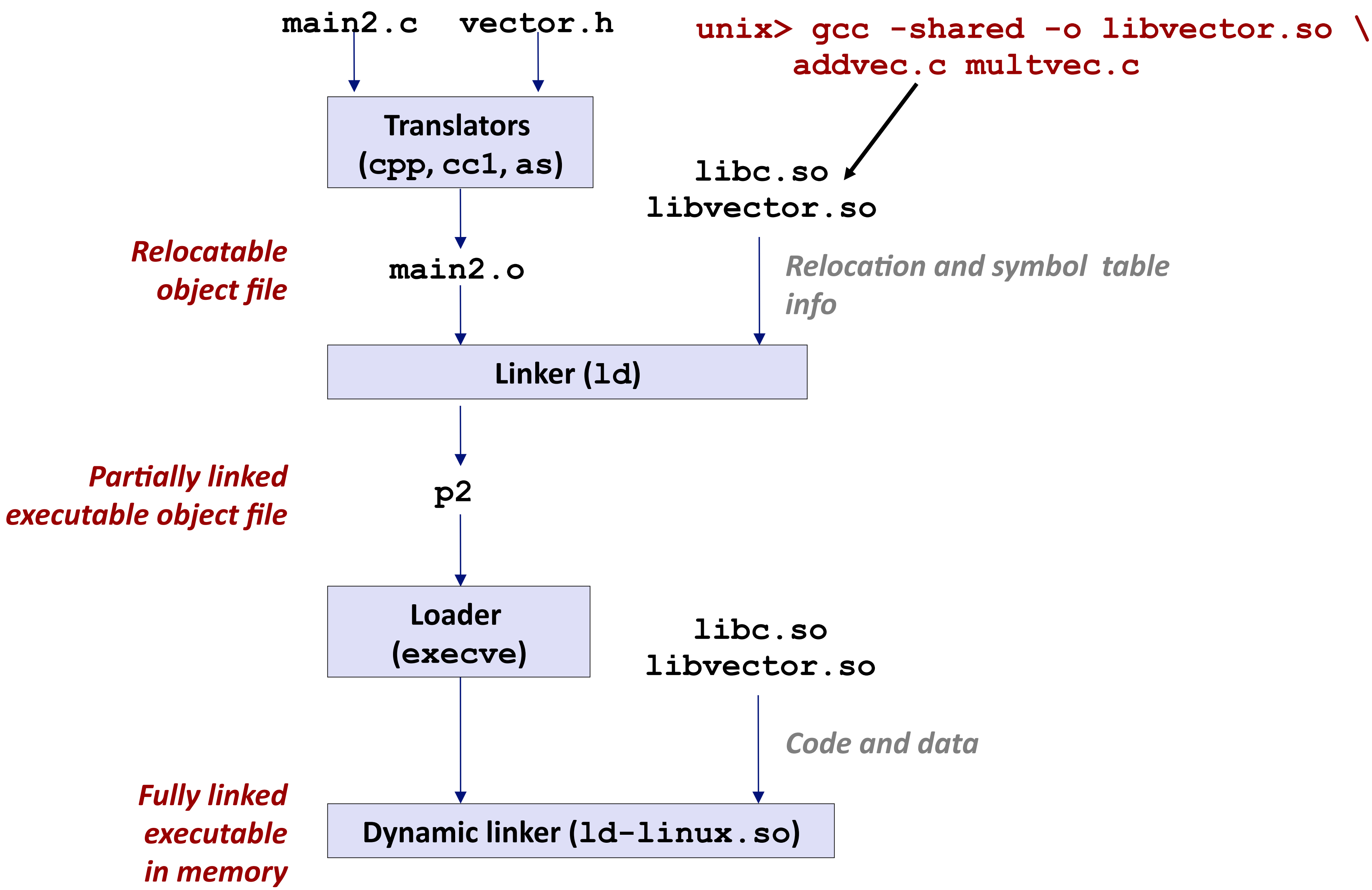
Instructors:

Dave O'Hallaron, Greg Ganger, and Greg Kesden

Today

- **Library interpositioning**
- Map-reduce
- Virtual Machines
- Cloud Computing

Dynamic Linking at Load-time (review)



Dynamic Linking at Run-time (review)

```
#include <stdio.h>
#include <dlfcn.h>

int x[2] = {1, 2};
int y[2] = {3, 4};
int z[2];

int main()
{
    void *handle;
    void (*addvec)(int *, int *, int *, int);
    char *error;

    /* Dynamically load the shared lib that contains addvec() */
    handle = dlopen("./libvector.so", RTLD_LAZY);
    if (!handle) {
        fprintf(stderr, "%s\n", dlerror());
        exit(1);
    }
```

Dynamic Linking at Run-time

```
    ...

    /* Get a pointer to the addvec() function we just loaded */
    addvec = dlsym(handle, "addvec");
    if ((error = dlerror()) != NULL) {
        fprintf(stderr, "%s\n", error);
        exit(1);
    }

    /* Now we can call addvec() just like any other function */
    addvec(x, y, z, 2);
    printf("z = [%d %d]\n", z[0], z[1]);

    /* unload the shared library */
    if (dlclose(handle) < 0) {
        fprintf(stderr, "%s\n", dlerror());
        exit(1);
    }
    return 0;
}
```

Case Study: Library Interpositioning

- ***Library interpositioning*** **: powerful linking technique that allows programmers to intercept calls to arbitrary functions**
- **Interpositioning can occur at:**
  - **Compile time:** When the source code is compiled
  - **Link time:** When the relocatable object files are statically linked to form an executable object file
  - **Load/run time:** When an executable object file is loaded into memory, dynamically linked, and then executed.

Some Interpositioning Applications

- **Security**
  - Confinement (sandboxing)
    - Interpose calls to libc functions.
  - Behind the scenes encryption
    - Automatically encrypt otherwise unencrypted network connections.
- **Monitoring and Profiling**
  - Count number of calls to functions
  - Characterize call sites and arguments to functions
  - Malloc tracing
    - Detecting memory leaks
    - **Generating address traces**

Example Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int main()
{
    free(malloc(10));
    printf("hello, world\n");
    exit(0);
}
```

hello.c

- **Goal: trace the addresses and sizes of the allocated and freed blocks, without modifying the source code.**
- **Three solutions: interpose on the lib `malloc` and `free` functions at compile time, link time, and load/run time.**

Compile-time Interpositioning

```
#ifdef COMPILETIME
/* Compile-time interposition of malloc and free using C
 * preprocessor. A local malloc.h file defines malloc (free)
 * as wrappers mymalloc (myfree) respectively.
 */

#include <stdio.h>
#include <malloc.h>

/*
 * mymalloc - malloc wrapper function
 */
void *mymalloc(size_t size, char *file, int line)
{
    void *ptr = malloc(size);
    printf("%s:%d: malloc(%d)=%p\n", file, line, (int)size,
ptr);
    return ptr;
}
```

mymalloc.c

Compile-time Interpositioning

```
#define malloc(size) mymalloc(size, __FILE__, __LINE__ )
#define free(ptr) myfree(ptr, __FILE__, __LINE__ )

void *mymalloc(size_t size, char *file, int line);
void myfree(void *ptr, char *file, int line);
```
malloc.h

```
linux> make helloc
gcc -O2 -Wall -DCOMPILETIME -c mymalloc.c
gcc -O2 -Wall -I. -o helloc hello.c mymalloc.o
linux> make runc
./helloc
hello.c:7: malloc(10)=0x501010
hello.c:7: free(0x501010)
hello, world
```

Link-time Interpositioning

```
#ifdef LINKTIME
/* Link-time interposition of malloc and free using the
static linker's (ld) "--wrap symbol" flag. */

#include <stdio.h>

void *__real_malloc(size_t size);
void __real_free(void *ptr);

/*
 * __wrap_malloc - malloc wrapper function
 */
void *__wrap_malloc(size_t size)
{
    void *ptr = __real_malloc(size);
    printf("malloc(%d) = %p\n", (int)size, ptr);
    return ptr;
}
```

mymalloc.c

Link-time Interpositioning

```
linux> make hellol
gcc -O2 -Wall -DLINKTIME -c mymalloc.c
gcc -O2 -Wall -Wl,--wrap,malloc -Wl,--wrap,free \
-o hellol hello.c mymalloc.o
linux> make runl
./hellol
malloc(10) = 0x501010
free(0x501010)
hello, world
```

- **The "`-Wl`" flag passes argument to linker**
- **Telling linker "`--wrap,malloc`" tells it to resolve references in a special way:**
  - Refs to `malloc` should be resolved as `__wrap_malloc`
  - Refs to `__real_malloc` should be resolved as `malloc`

```c
#ifdef RUNTIME
 /* Run-time interposition of malloc and free based on
 * dynamic linker's (ld-linux.so) LD_PRELOAD mechanism */
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

void *malloc(size_t size)
{
    static void *(*mallocp)(size_t size);
    char *error;
    void *ptr;

    /* get address of libc malloc */
    if (!mallocp) {
        mallocp = dlsym(RTLD_NEXT, "malloc");
        if ((error = dlerror()) != NULL) {
            fputs(error, stderr);
            exit(1);
        }
    }
    ptr = mallocp(size);
    printf("malloc(%d) = %p\n", (int)size, ptr);
    return ptr;
}
```

mymalloc.c

Load/Run-time Interpositioning

Load/Run-time Interpositioning

```
linux> make hellor
gcc -O2 -Wall -DRUNTIME -shared -fPIC -o mymalloc.so mymalloc.c
gcc -O2 -Wall -o hellor hello.c
linux> make runr
(LD_PRELOAD="/usr/lib64/libdl.so ./mymalloc.so" ./hellor)
malloc(10) = 0x501010
free(0x501010)
hello, world
```

- **The `LD_PRELOAD` environment variable tells the dynamic linker to resolve unresolved refs (e.g., to `malloc`) by looking in `libdl.so` and `mymalloc.so` first.**
  - `libdl.so` necessary to resolve references to the `dlopen` functions.

Interpositioning Recap

- **Compile Time**
  - Apparent calls to `malloc`/`free` get macro-expanded into calls to `mymalloc`/`myfree`
- **Link Time**
  - Use linker trick to have special name resolutions
    - `malloc` → `__wrap_malloc`
    - `__real_malloc` → `malloc`
- **Load/Run Time**
  - Implement custom version of `malloc`/`free` that use dynamic linking to load library `malloc`/`free` under different names

Today

- Library interpositioning
- **Map-reduce**
- Virtual Machines
- Cloud Computing

Parallel Programming Building Blocks

- **Not usually done fully "by hand"**
  - Major parallel programming exploits building blocks
  - For programming efficiency and portability
- **Example: OpenMP**
  - API and framework for parallel execution
  - for "shared memory" parallel programming
    - such as many-core systems
- **Example: MPI (Message Passing Interface)**
  - API and middleware for multi-machine parallel execution
- **Example: OpenGL**
  - API and framework for high-performance graphics
  - includes mapping to popular graphics accelerators and "GPUs"
- **Example: Map-Reduce…**

Map-Reduce Programming

- **Easy-to-use API for data-parallel programs**
  - "data-parallel" means that different data processed in parallel
    - by the same sub-program
  - partial results can then be combined
- **Programmer writes two functions**
  - Map(k1, v1): outputs a list of [k2, v2] pairs
    - common (but not required) for map functions to filter the input
  - Reduce(k2, list of v2 values): outputs a list of values (call it v3)
- **Easy to make parallel**
  - Map instances can execute in any order
  - Reduce instances can execute in any order (after all maps finish)
- **Described by a 2004 Google paper**
  - Used extensively by Google, Facebook, Twitter, etc.
  - Most use the open source (Apache) implementation called Hadoop

M-R Example: Word Frequency in Web Pages

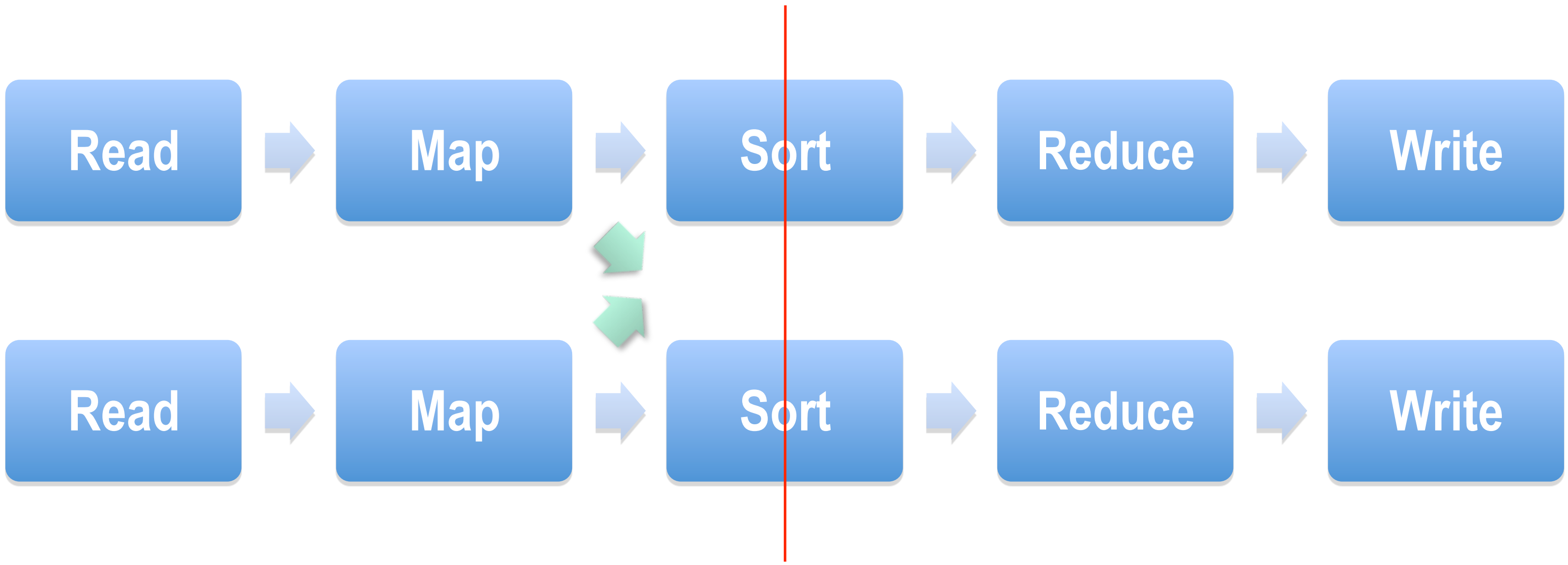
```
void map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
     EmitIntermediate(w, "1");
```

- **Input and output Strings**
  - Java pseudo-code here
- **Map breaks out each word**
- **Reduce counts occurrences**
  - Iterator provides the value list

```
void reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  int sum = 0;
  for each pc in partialCounts:
     sum += ParseInt(pc);
  Emit(word, AsString(sum));
```

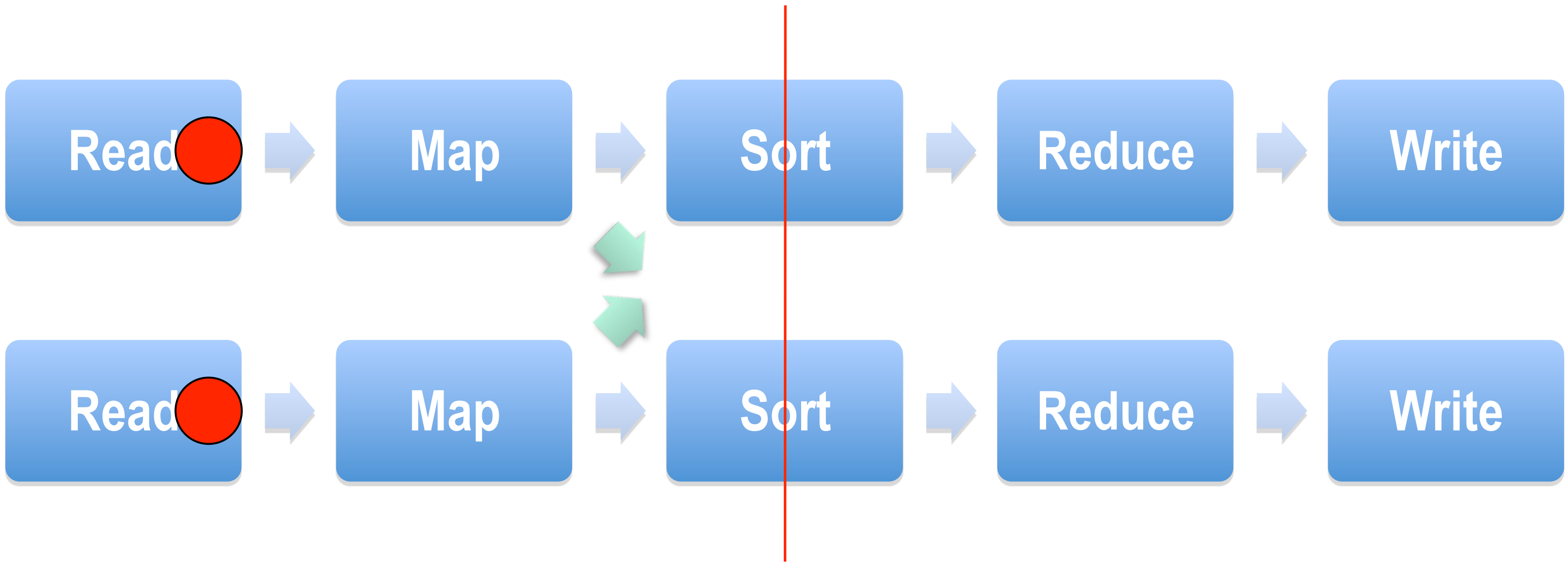
Visual of a Map-reduce Dataflow

Phase 1: read, map and shuffle data



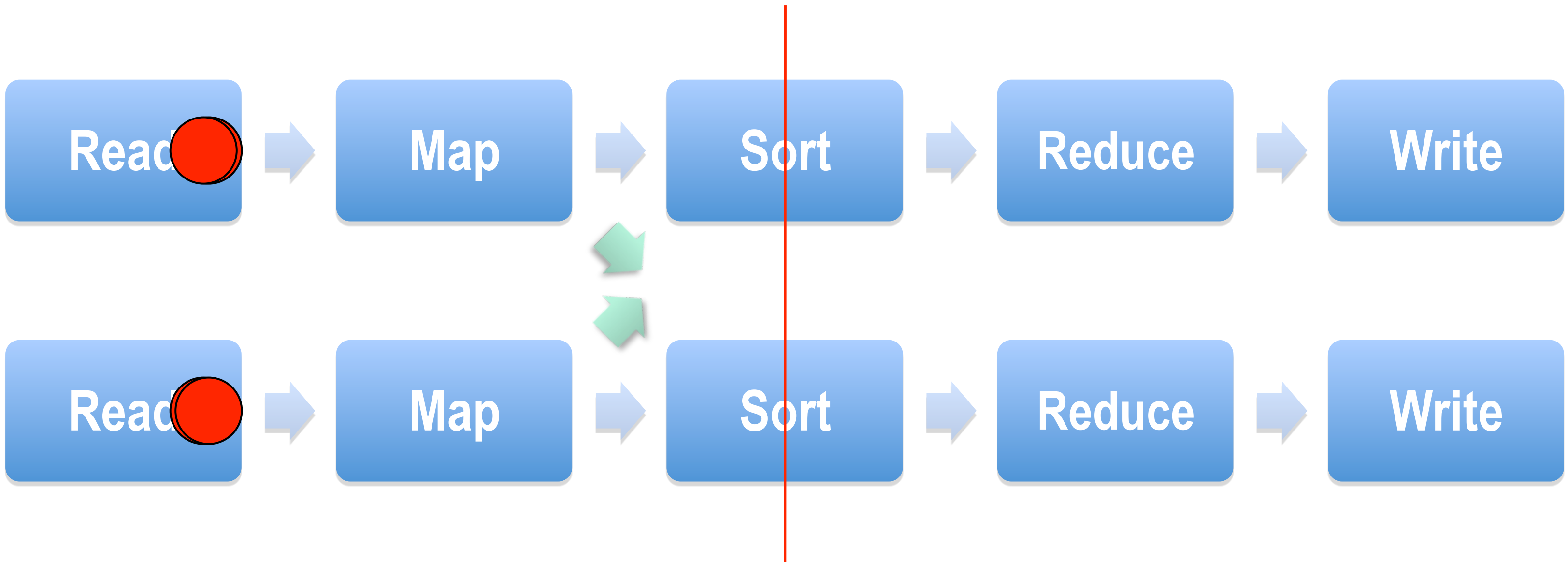
Visual of a Map-reduce Dataflow

Phase 1: read, map and shuffle data



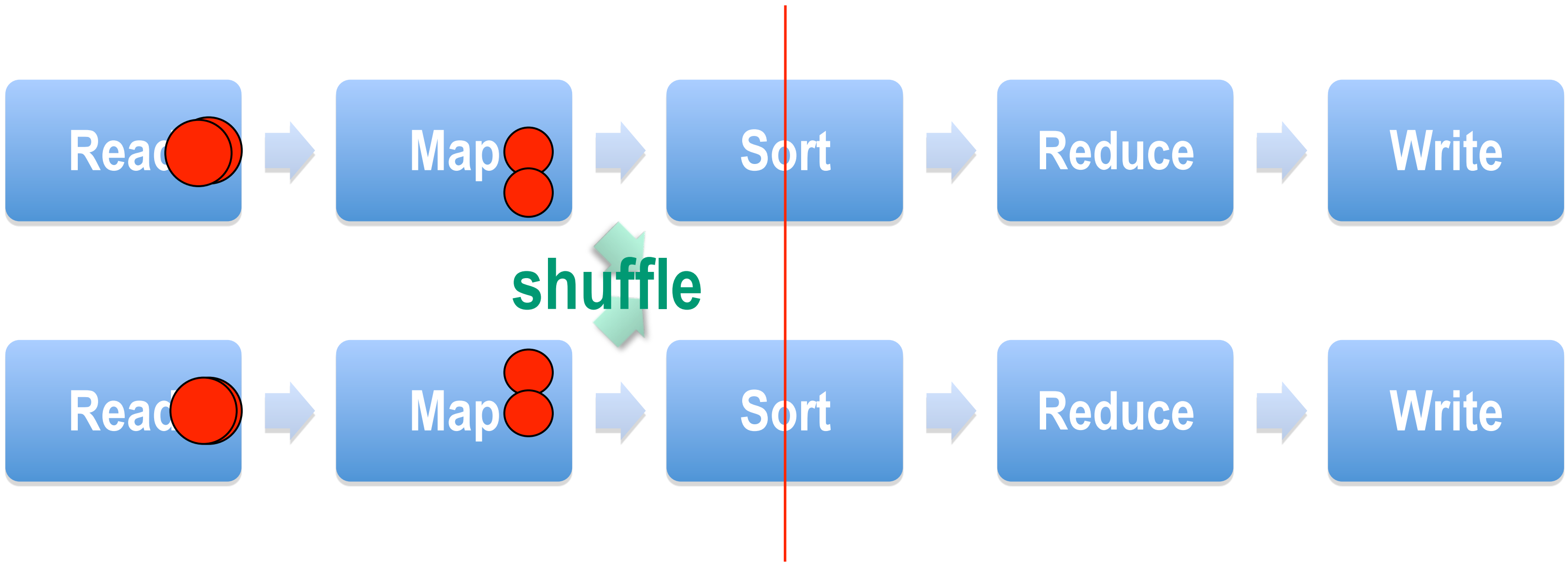
Visual of a Map-reduce Dataflow

Phase 1: read, map and shuffle data



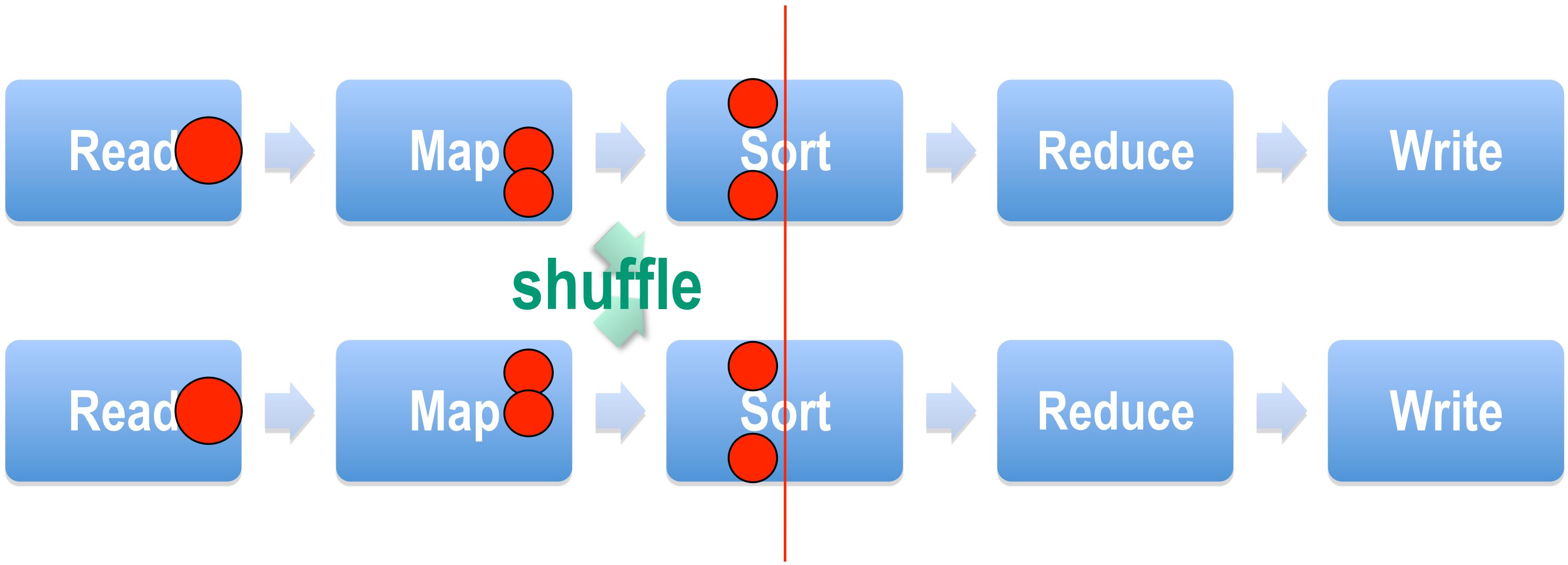
Visual of a Map-reduce Dataflow

Phase 1: read, map and shuffle data



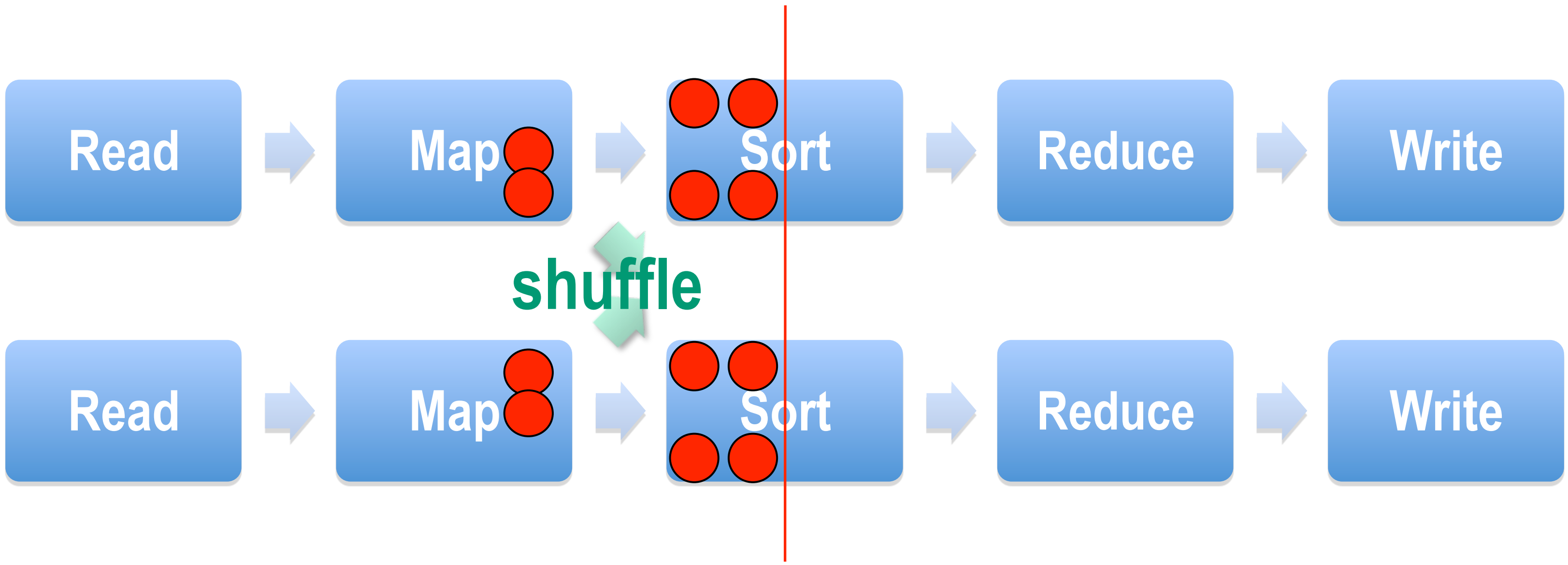
Visual of a Map-reduce Dataflow

Phase 1: read, map and shuffle data



Visual of a Map-reduce Dataflow

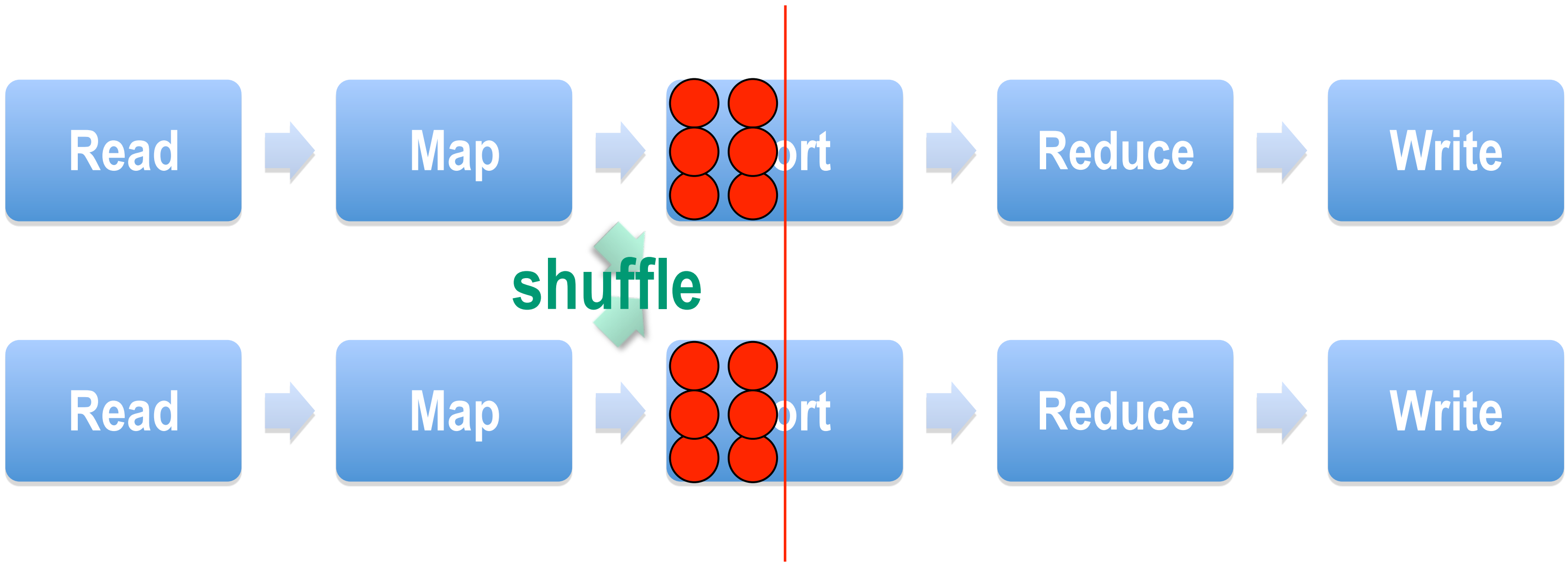
Phase 1: read, map and shuffle data



- Sort introduces barrier that disrupts pipeline

Visual of a Map-reduce Dataflow

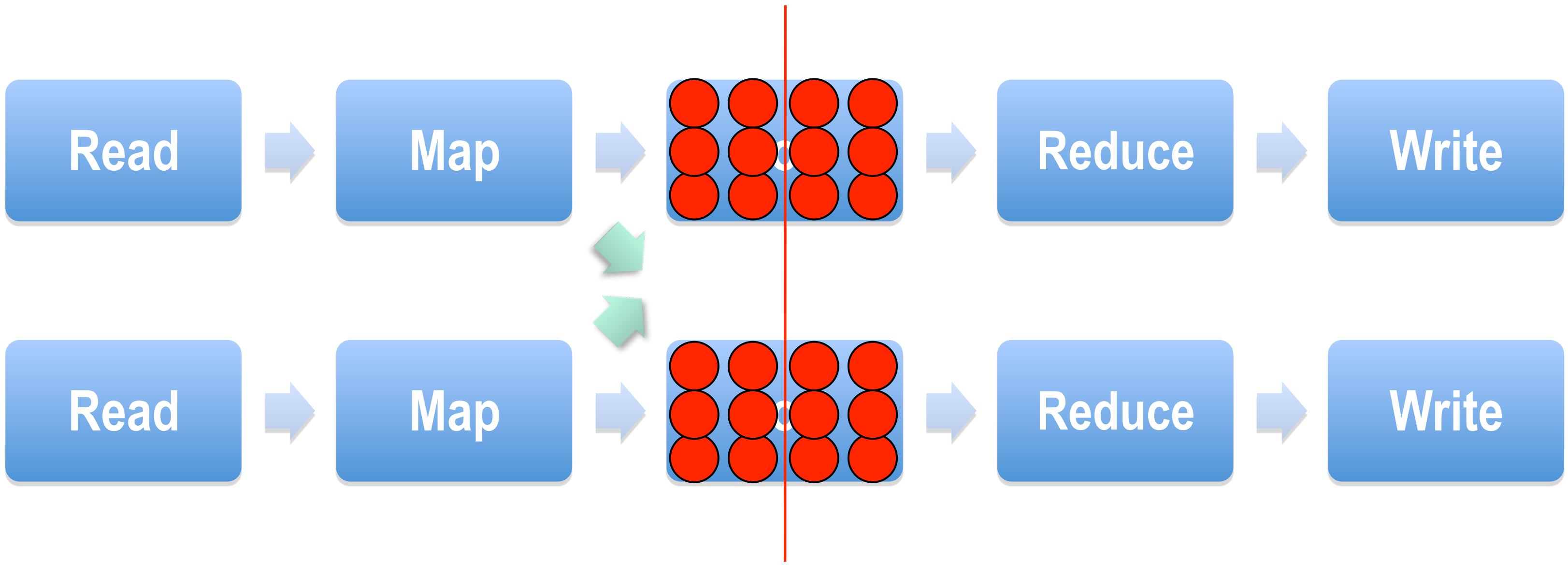
Phase 1: read, map and shuffle data



- Sort introduces barrier that disrupts pipeline

Visual of a Map-reduce Dataflow

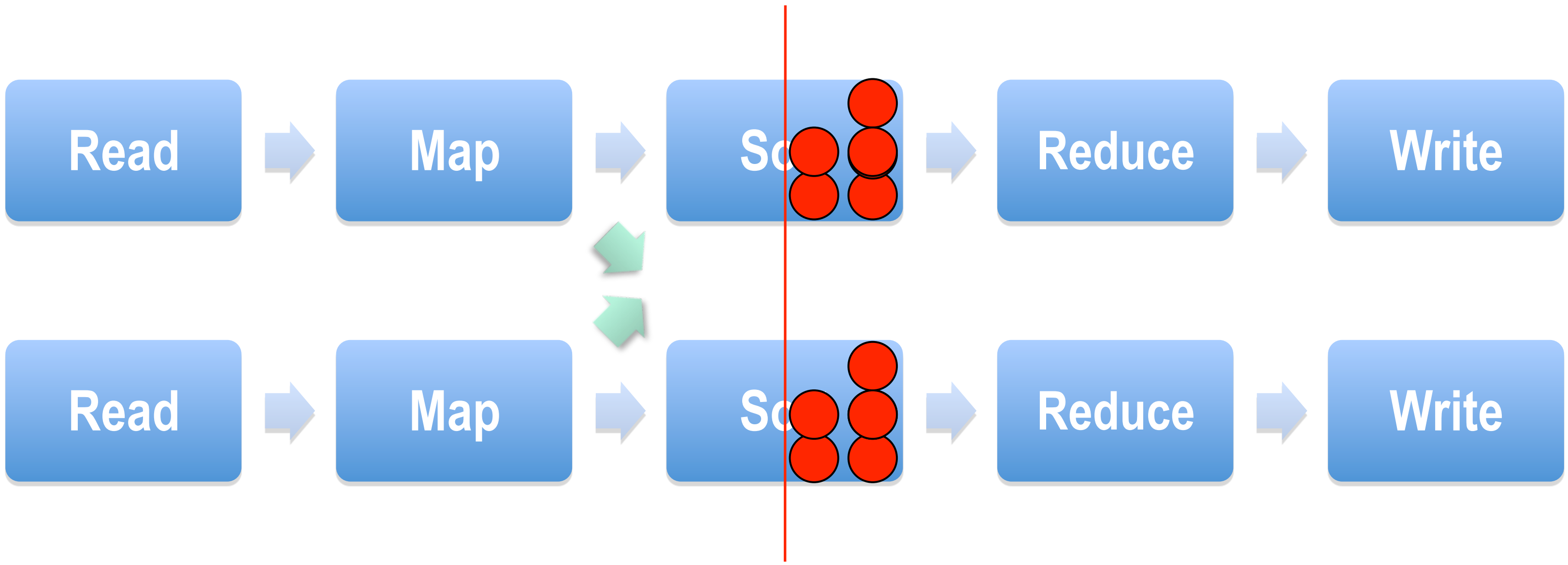
Phase 2: sort, reduce, and write data



- Sort introduces barrier that disrupts pipeline

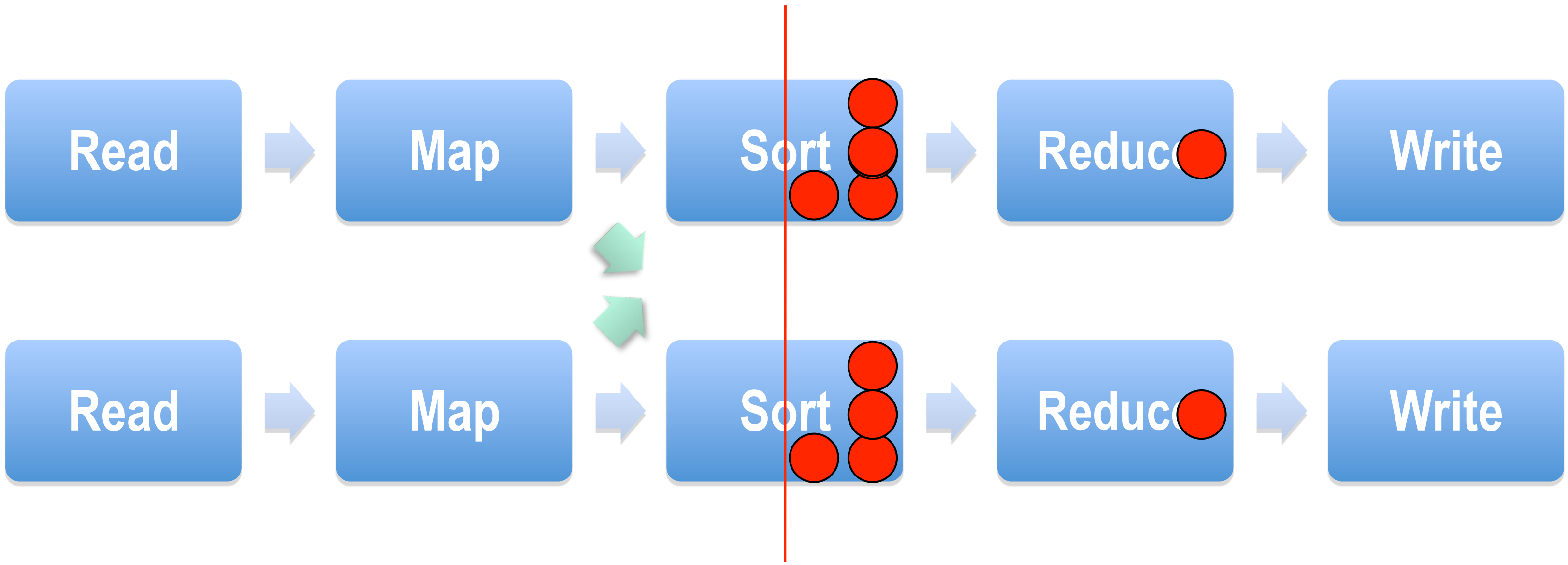
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



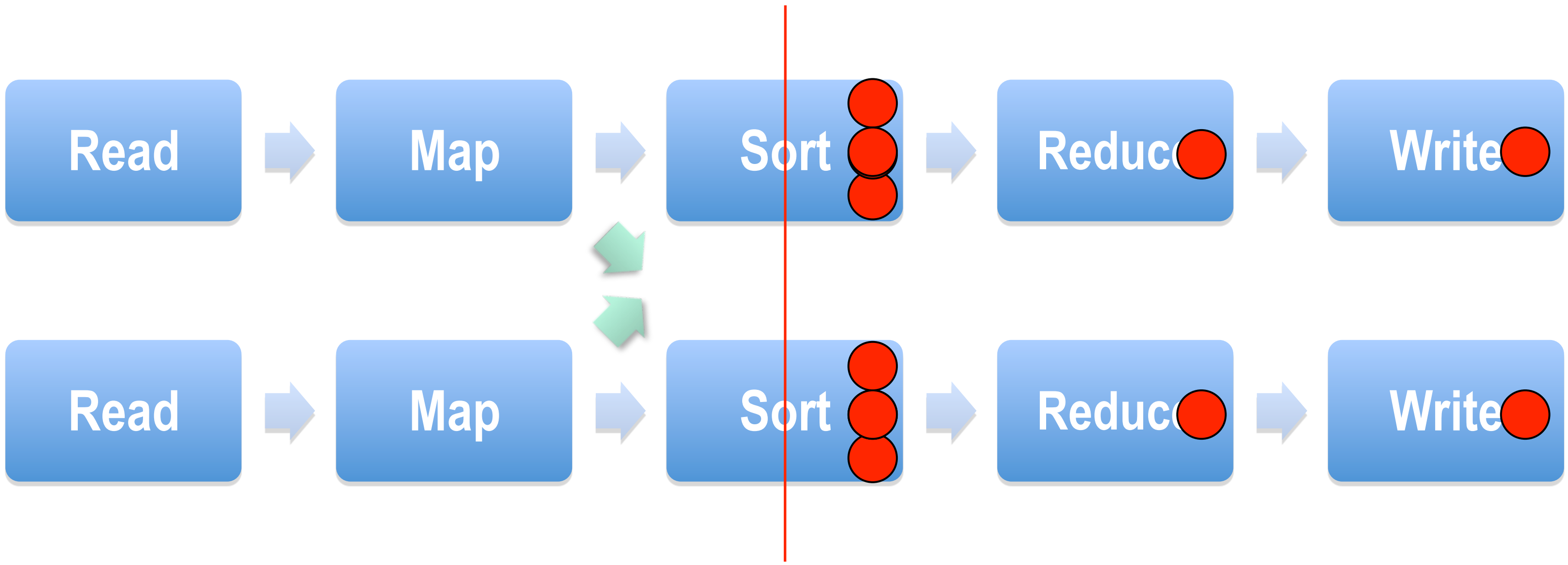
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



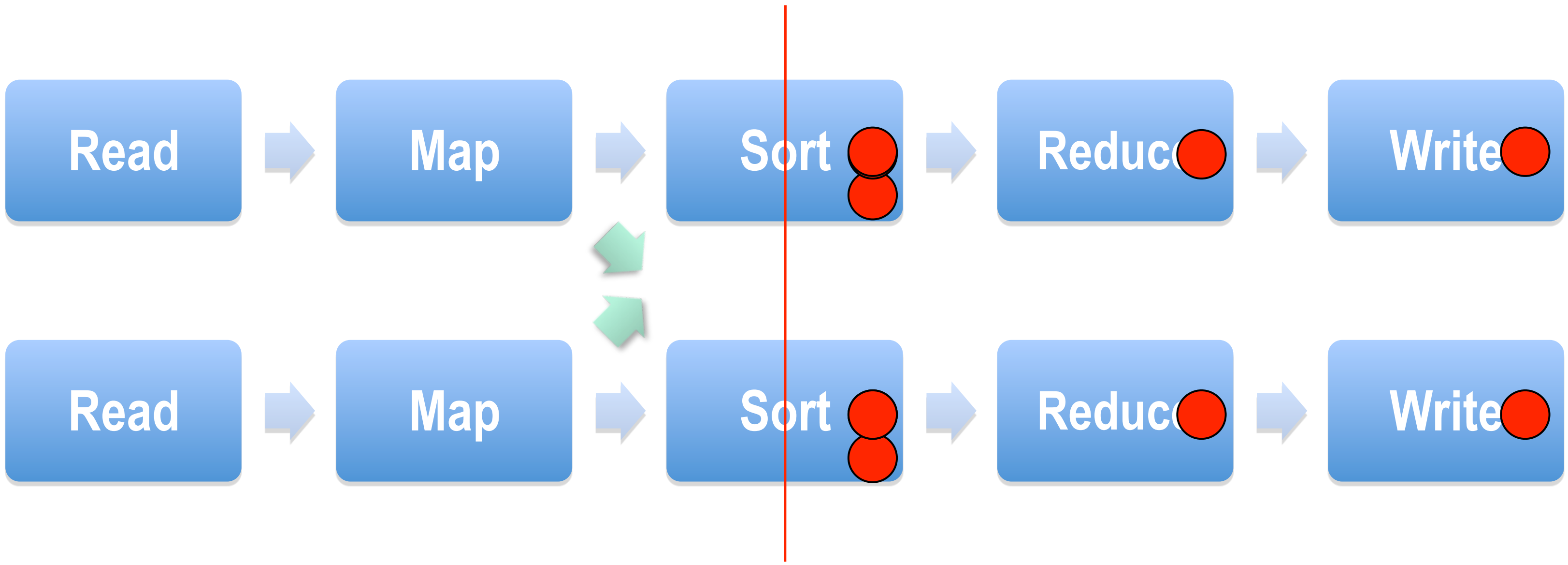
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



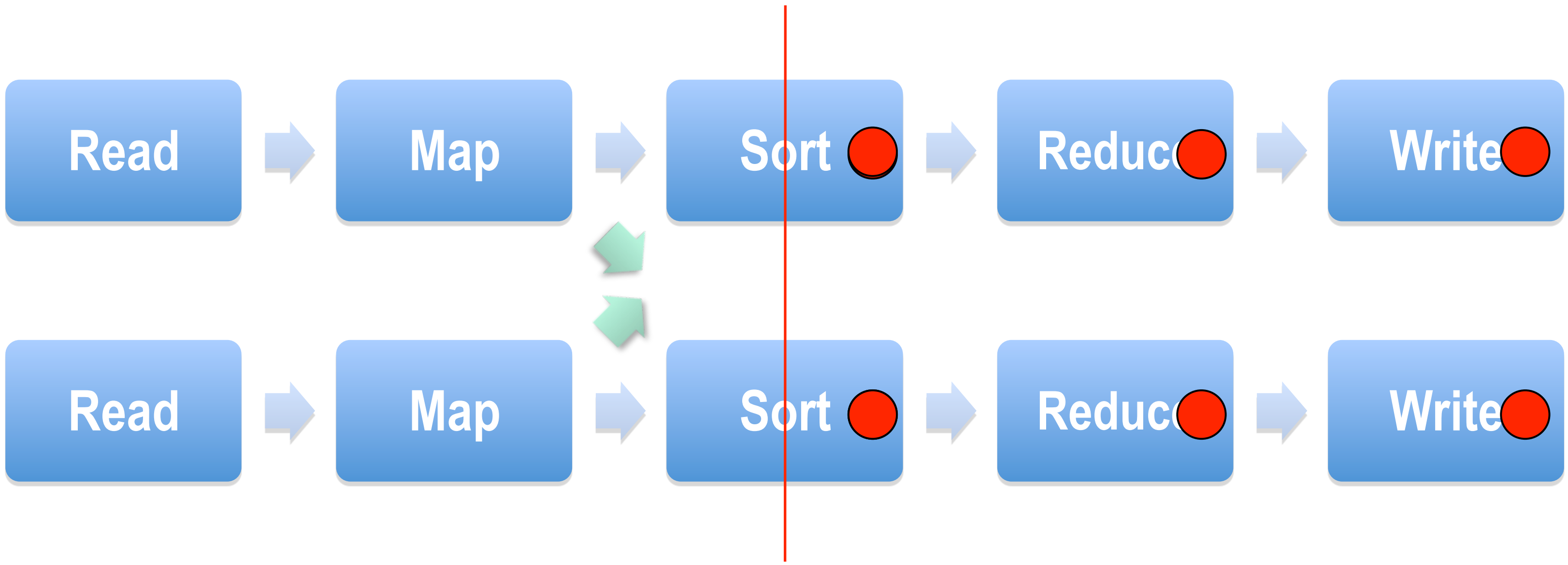
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



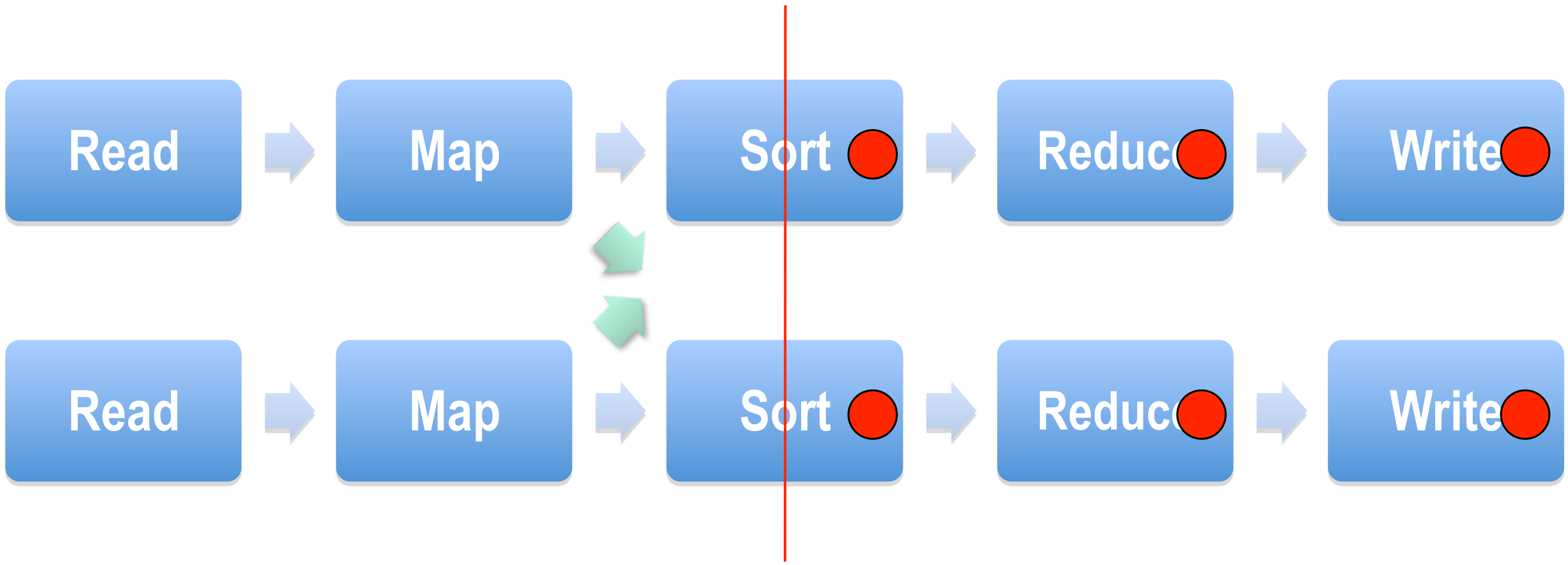
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



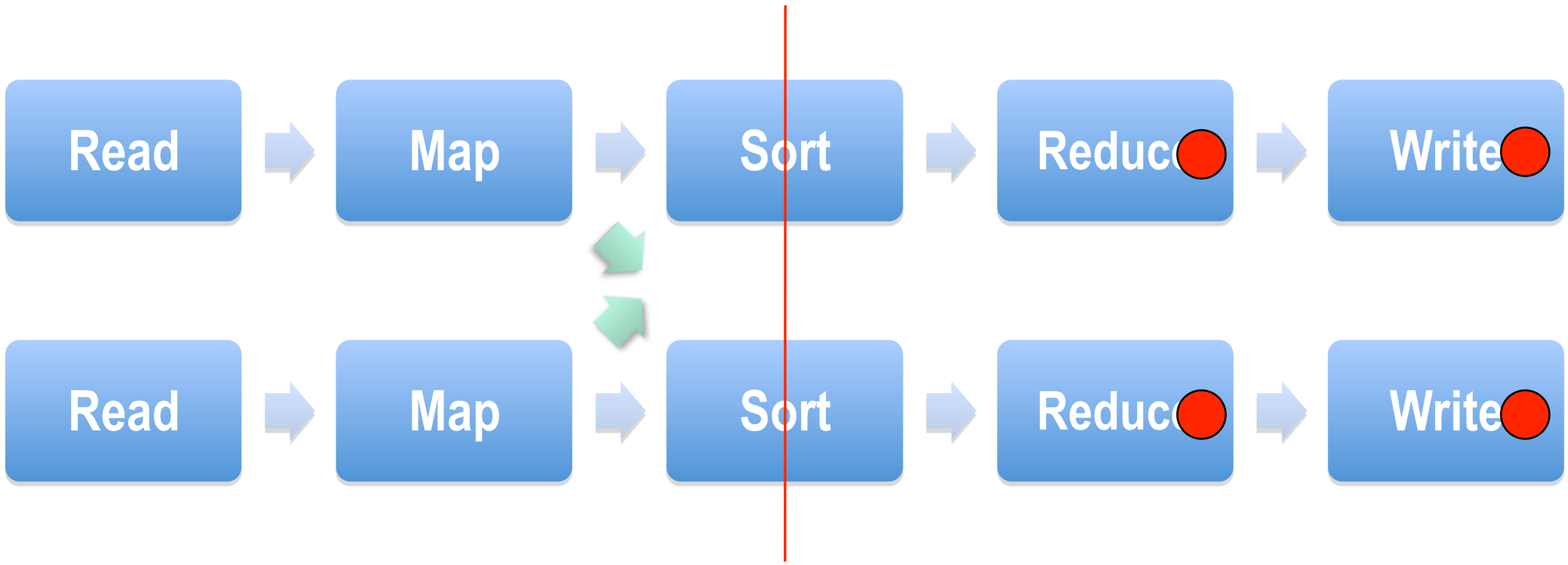
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



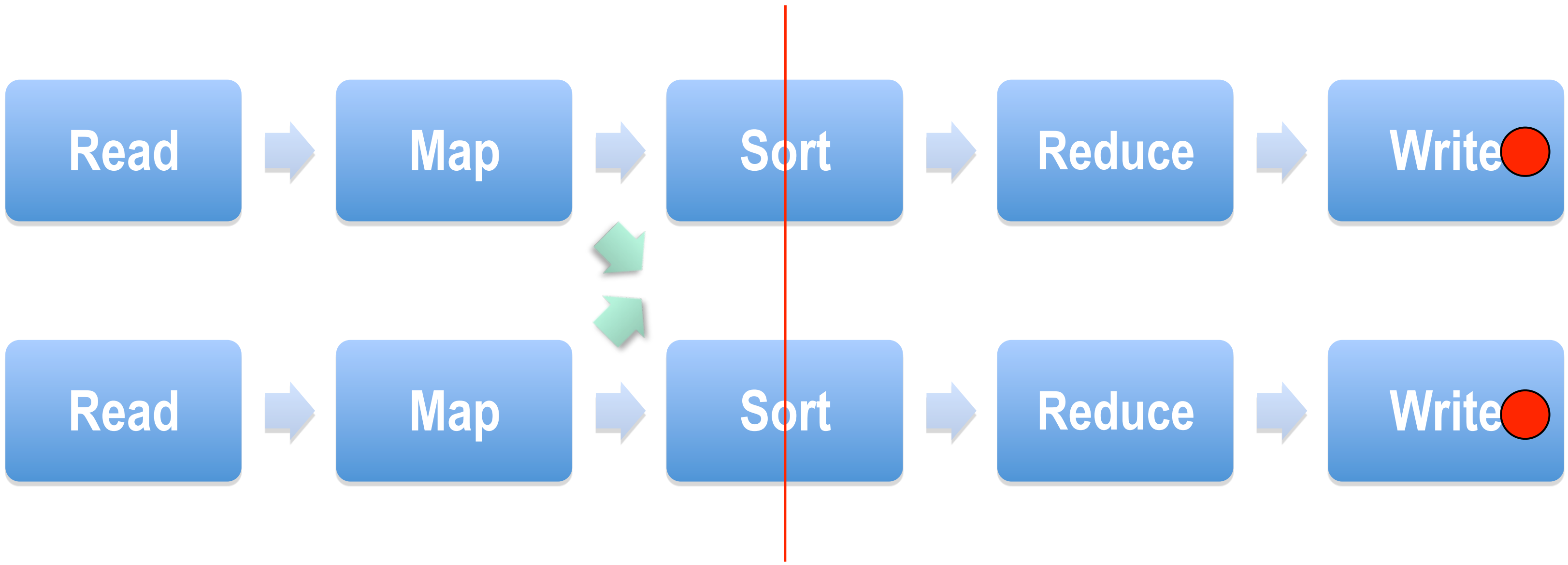
Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



Visual of a Map-reduce Dataflow

Phase 2: sort, reduce, and write data



Comments on Map-reduce

- **Effective at large scale**
  - Google and others use it across 1000s of machines and PBs of data
    - to generate search indices, translate languages, and many other things
  - Used for setting sort benchmark records (e.g., TeraSort and PetaSort)
- **Indirectly helped spawn shift toward Data-Intensive Computing**
  - in which insights are mined from lots of observation data
  - Search for "Unreasonable Effectiveness of Data"
- **Not the "be all / end all" for parallel programming**
  - Great for relatively simple data-parallel activities
    - e.g., sifting through huge amounts of data
  - Not great for advanced machine learning algorithms
    - so, even newer APIs/frameworks being developed to support those

Today

- Library interpositioning
- Map-reduce
- **Virtual Machines**
- Cloud Computing

Virtual Machines

- **Decouple physical HW reality from exposed view**
  - We've seen "virtual memory" and processes
  - Apply same concept more generally
    - "virtual disks", "virtual networks", "virtual machines", etc.
- **Why virtual machines?**
  - Flexibility
  - Efficiency
  - Security
- **Virtual machines (VMs) are increasingly common**
  - Linux KVM, VirtualBox, Xen, Vmware, MS Virtual Server
  - Autolab autograding backend uses VMs
  - Enable cloud computing:
    - Proprietary cloud services: EC2, Rackspace, Compute Engine
    - Open source cloud system: OpenStack

Today

- Library interpositioning
- Map-reduce
- Virtual Machines
- **Cloud Computing**

What is Cloud Computing?

- **Short version:**
  - Using someone else's computers (and maybe software)
    - instead of buying/maintaining one's own
    - elastic and on-demand (pay for what need)
  - Sharing those computers with other "tenants"
    - instead of having them all-to-oneself
- **Longer version:**
  - See NIST's more complex definition (2 pages!)
    - a more technical and comprehensive statement
    - notes multiple styles, along multiple dimensions

Why Cloud Computing?

- **Huge potential benefits**
  - Consolidation
    - Higher server utilization (7-25% -> 70+%)
    - Economies of scale
    - E.g., HP went from 80+ data centers to 6
      - and saved $1B/year… over 60% of total annual expense
  - Aggregation
    - One set of experts doing it for many
      - Instead of each for themselves
- **Rapid deployment**
  - Rent when ready and scale as need
    - Rather than specify, buy, deploy, setup, then start

3 Styles of Cloud Computing

- **IaaS – Infrastructure as a Service**
  - Data center rents VMs to users
    - Ex: Amazon EC2
  - User must install SW (platform & application)
- **PaaS – Platform as a Service**
  - Offer ready-to-run platform solutions
    - Ex: Google App Engine, Microsoft Azure
  - User develops/installs applications
- **SaaS – Software as a Service**
  - Complete application solutions are offered
  - Ex: Gmail, Salesforce.com, etc.

Cloud Computing Accessibility

- **Private vs. Public Clouds**
  - Private cloud: one organization
    - Multiple groups sharing a common infrastructure
    - Incredibly popular in business world, right now
  - Public cloud: many organizations
    - e.g., Internet offerings

Deeper: Operational Costs Out of Control

- **Power and cooling**
  - Now on par with purchase costs
  - Trends making it worse every year
    - Power/heat go up with speed
    - Cluster sizes increase due to commodity pricing

EPA report about 2011 data center power usage:

In 2006, 1.5% of total U.S. electricity consumption

"Under current efficiency trends, national energy consumption by servers and data centers could nearly double again in another five years (i.e., by 2011) to more than 100 billion kWh."

[i.e., 2-3% of total U.S. consumption]

A few "fun" data center energy facts

"Google's power consumption ... would incur an annual electricity bill of nearly $38 million"

[Qureshi:sigcomm09]

"Energy consumption by … data centers could nearly double ... (by 2011) to more than 100 billion kWh, representing a $7.4 billion annual electricity cost"

[EPA Report 2007]

Annual cost of energy for Google, Amazon, Microsoft

=

Annual cost of all first-year CS PhD Students

Deeper: Operational Costs Out of Control

- **Power and cooling**
  - Now on par with purchase costs
  - Trends making it worse every year
    - Power/heat go up with speed
    - Cluster sizes increase due to commodity pricing
- **Administration costs**
  - Often reported at 4-7X capital expenditures
  - Trends making it worse every year
    - Complexity goes up with features, expectations and cluster size
    - Salaries go up while equipment costs go down

Thanks!