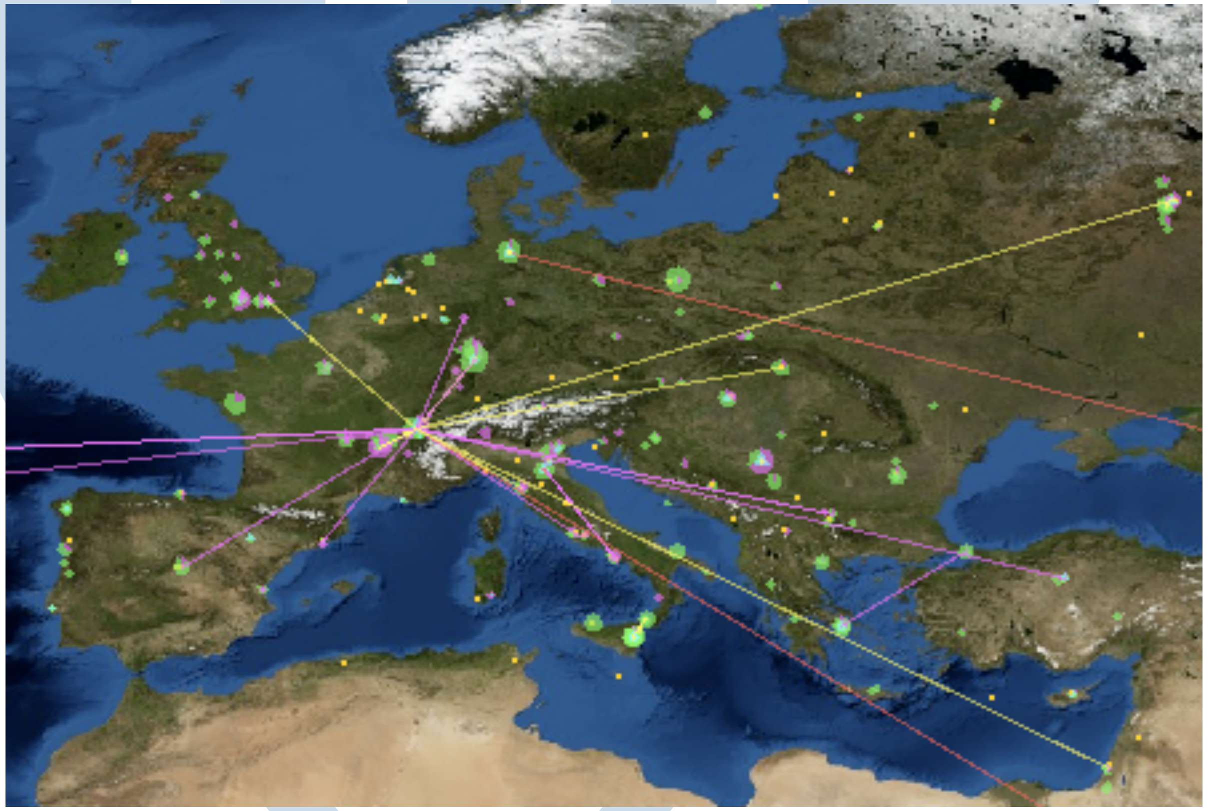
Migrating large output applications to Grid environments:
a simple library for threaded transfers with gLite

GoLP/IPFN Instituto Superior Técnico

Paulo Abreu

Ricardo Fonseca
(GoLP/IST, DCTI/ISCTE)

Luís O. Silva
(GoLP/IST)



GoLP/Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico (IST)
Lisbon, Portugal
http://cfp.ist.utl.pt/golp

Outline

- Introduction:
  - motivation & goals;
  - existing technologies in gLite.
- Deployment:
  - algorithm;
  - application deployment;
  - results.
- Conclusions:
  - ongoing work;
  - future directions.

Motivation

Motivation

- **int.eu.grid is here:**
  - parallel computational resources in Grid environment.

Motivation

- **int.eu.grid is here:**
  - parallel computational resources in Grid environment.
- **Deployment of HPC parallel codes:**
  - we develop & maintain 3 different massively parallel plasma simulation codes.

Motivation

- **int.eu.grid is here:**
  - parallel computational resources in Grid environment.
- **Deployment of HPC parallel codes:**
  - we develop & maintain 3 different massively parallel plasma simulation codes.
- **Main challenge: use Grid storage:**
  - small input data, very large output data;
  - **total output might not fit** in a WN;
  - **output available ASAP for post-processing** (eg., visualization).

Goals

Goals

User transparency:

- no difference between the local version and the Grid version;
- minimal performance impact;
- results are available as soon as they are produced.

Goals

User transparency:

- no difference between the local version and the Grid version;
- minimal performance impact;
- results are available as soon as they are produced.

Developer point of view:

- minimal integration effort,
- independent of output data format.

Existing technologies in gLite

	Advantages	Disadvantages
FTS		
GFAL		
LCG Utils		

Existing technologies in gLite

	Advantages	Disadvantages
FTS	◉ reliable, ◉ asynchronous, ◉ file to file;	
GFAL		
LCG Utils		

Existing technologies in gLite

	Advantages	Disadvantages
FTS	reliable, asynchronous, file to file;	scheduled transfers, low level, complexity.
GFAL		
LCG Utils		

Existing technologies in gLite

	Advantages	Disadvantages
FTS	◉ reliable, ◉ asynchronous, ◉ file to file;	◉ scheduled transfers, ◉ low level, ◉ complexity.
GFAL	◉ high level, ◉ POSIX-like, ◉ direct access;	
LCG Utils		

Existing technologies in gLite

	Advantages	Disadvantages
FTS	reliable, asynchronous, file to file;	scheduled transfers, low level, complexity.
GFAL	high level, POSIX-like, direct access;	data format dependent, memory usage, application stall.
LCG Utils		

Existing technologies in gLite

	Advantages	Disadvantages
FTS	reliable, asynchronous, file to file;	scheduled transfers, low level, complexity.
GFAL	high level, POSIX-like, direct access;	data format dependent, memory usage, application stall.
LCG Utils	high level, file to file;	

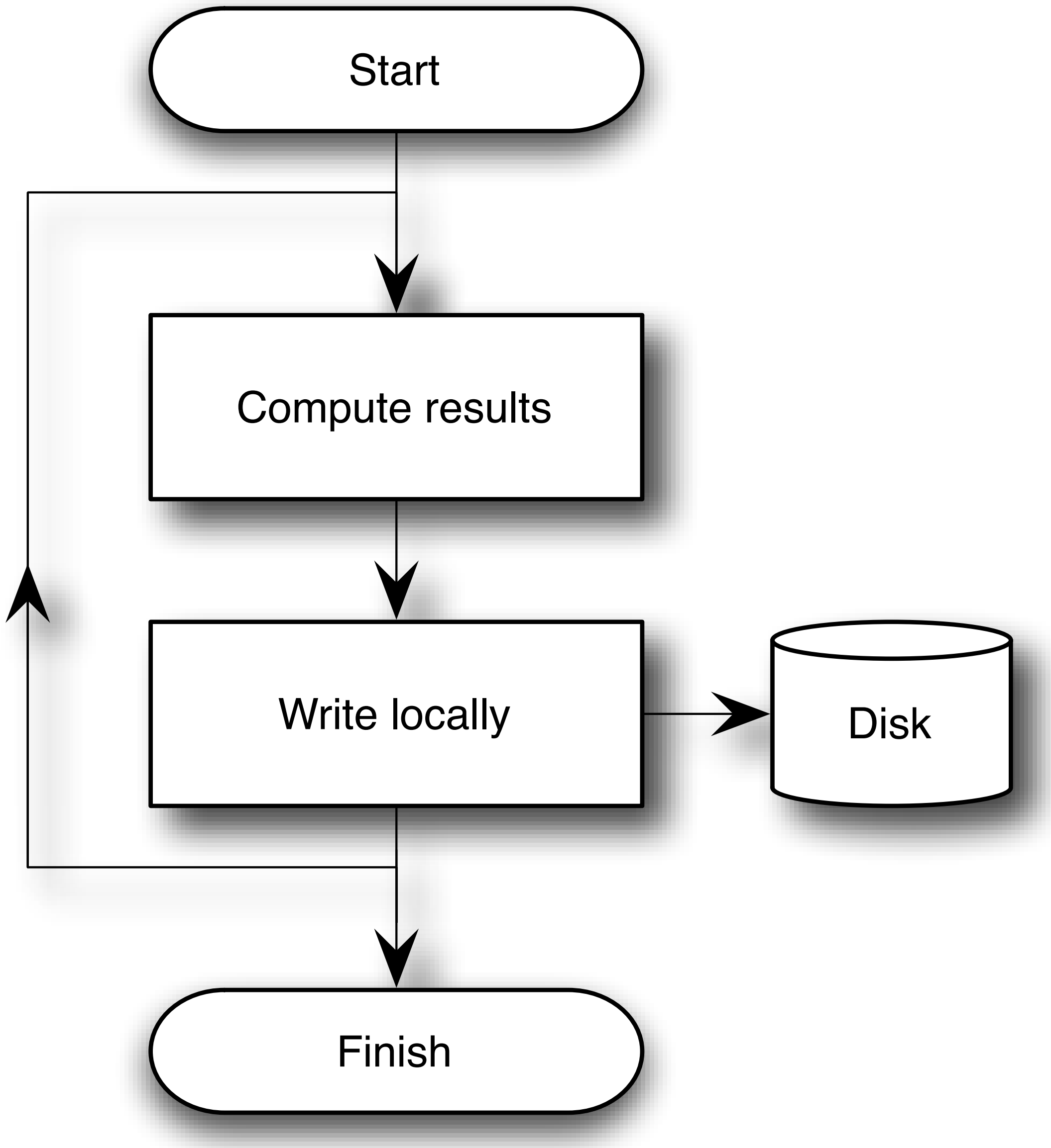
Existing technologies in gLite

	Advantages	Disadvantages
FTS	• reliable, • asynchronous, • file to file;	• scheduled transfers, • low level, • complexity.
GFAL	• high level, • POSIX-like, • direct access;	• data format dependent, • memory usage, • application stall.
LCG Utils	• high level, • file to file;	• high overhead, • configuration options.

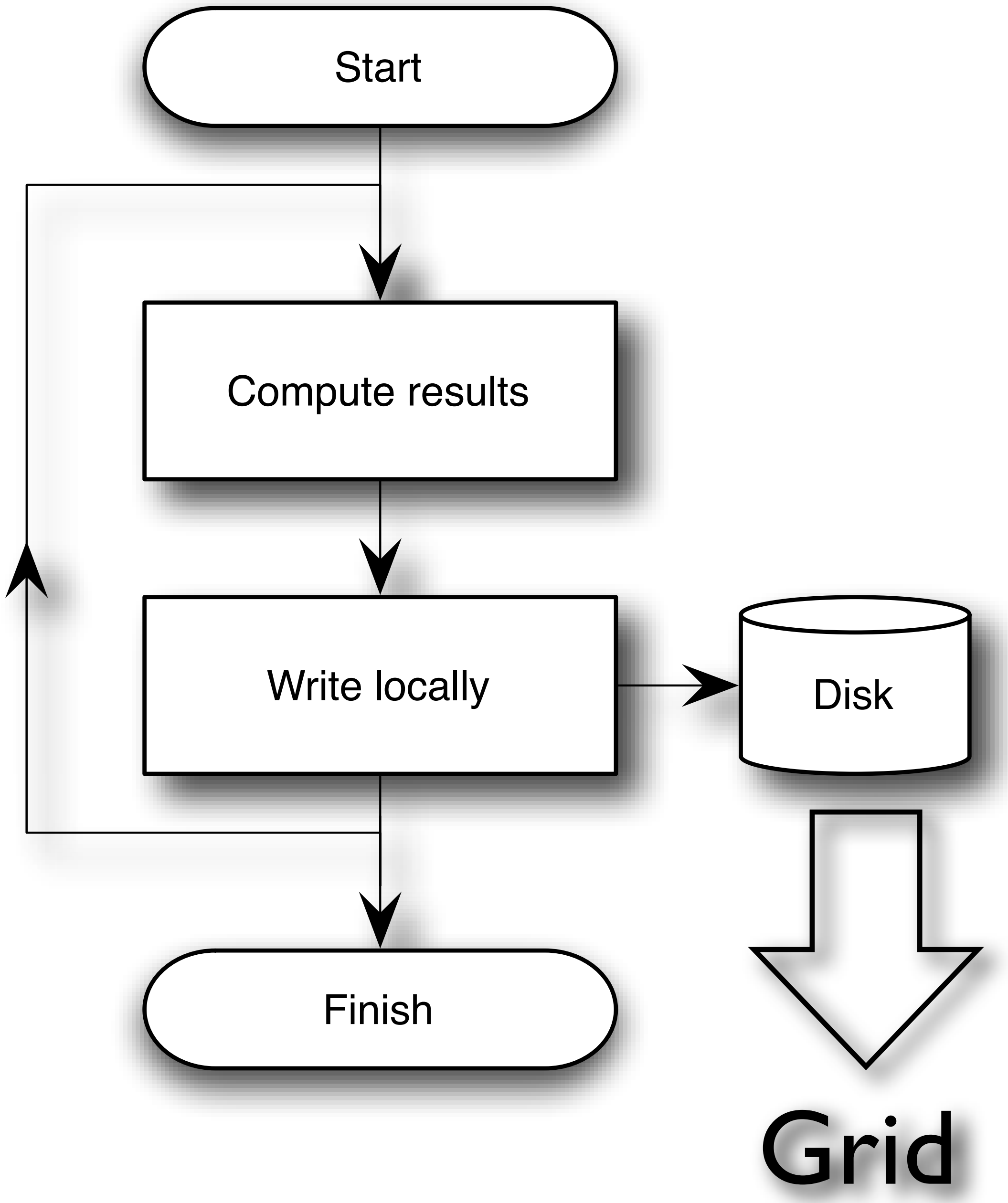
Existing technologies in gLite

	Advantages	Disadvantages
FTS	reliable, asynchronous, file to file;	scheduled transfers, low level, complexity.
GFAL	high level, POSIX-like, direct access;	data format dependent, memory usage, application stall.
LCG Utils	high level, file to file;	high overhead, configuration options.

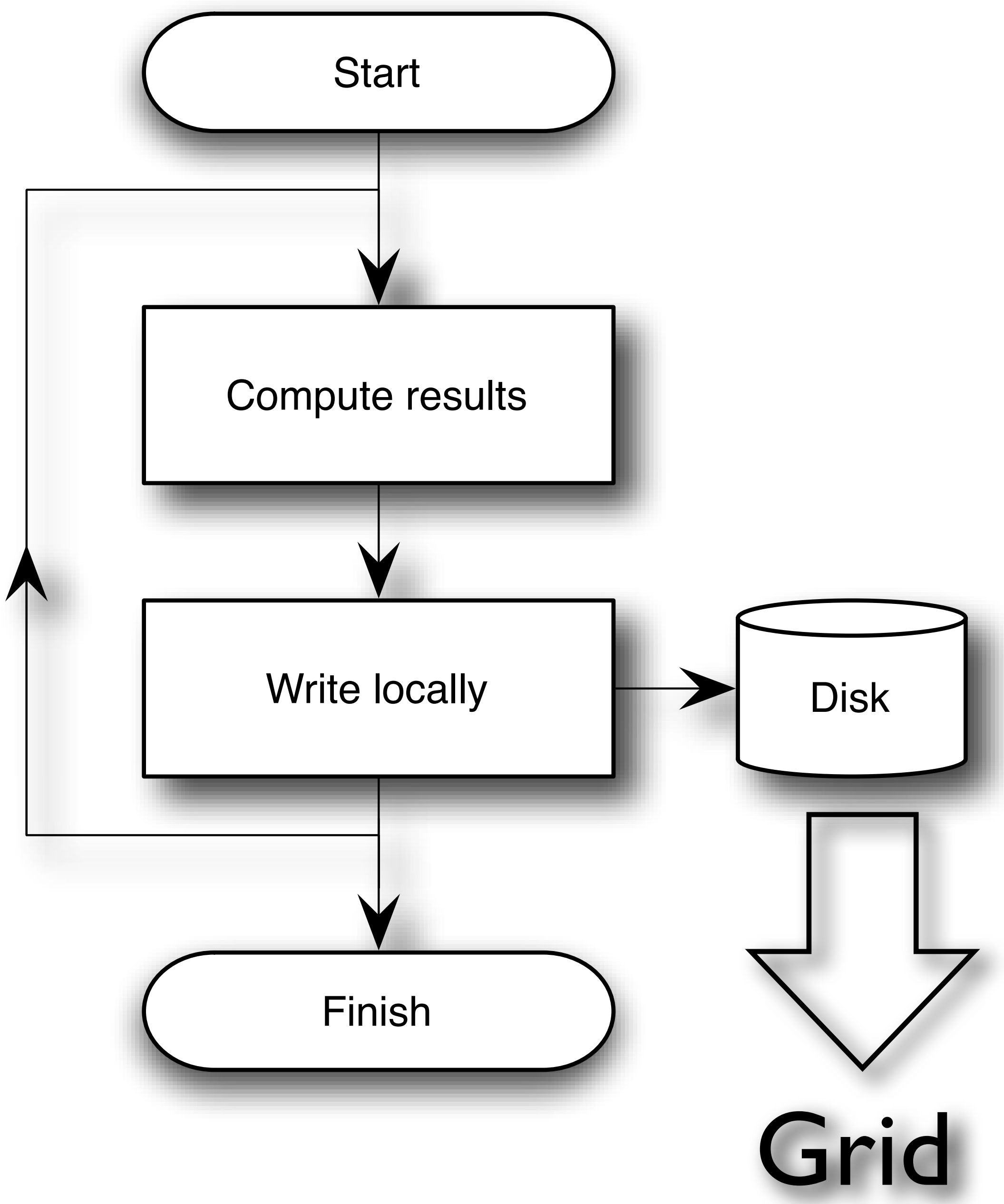
Data flow



Data flow



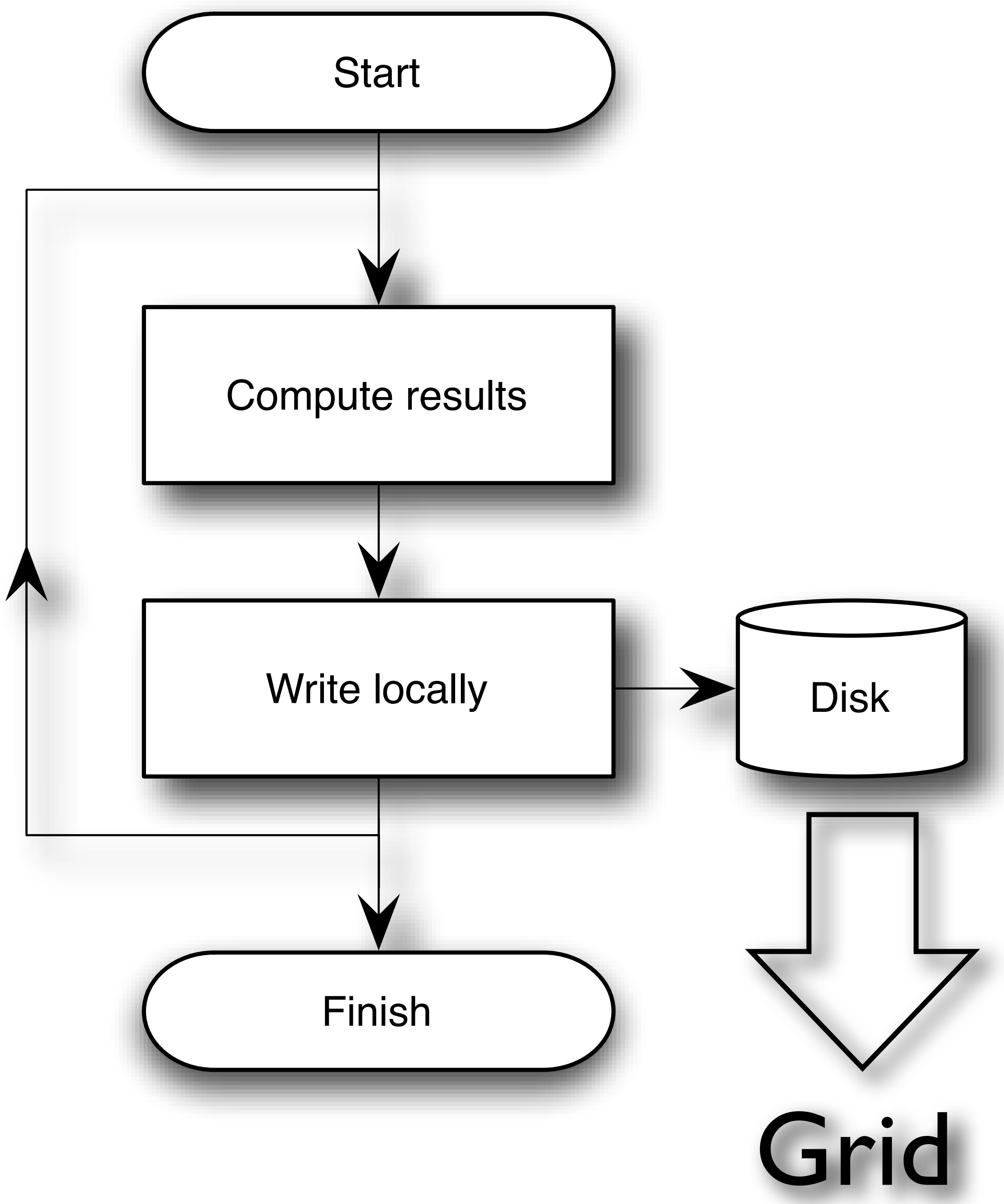
Data flow



Measurable time intervals:

- total turn-over time *T*,
- first turn-over time *F*,
- simulation step time *S*,
- complete simulation time *C*.

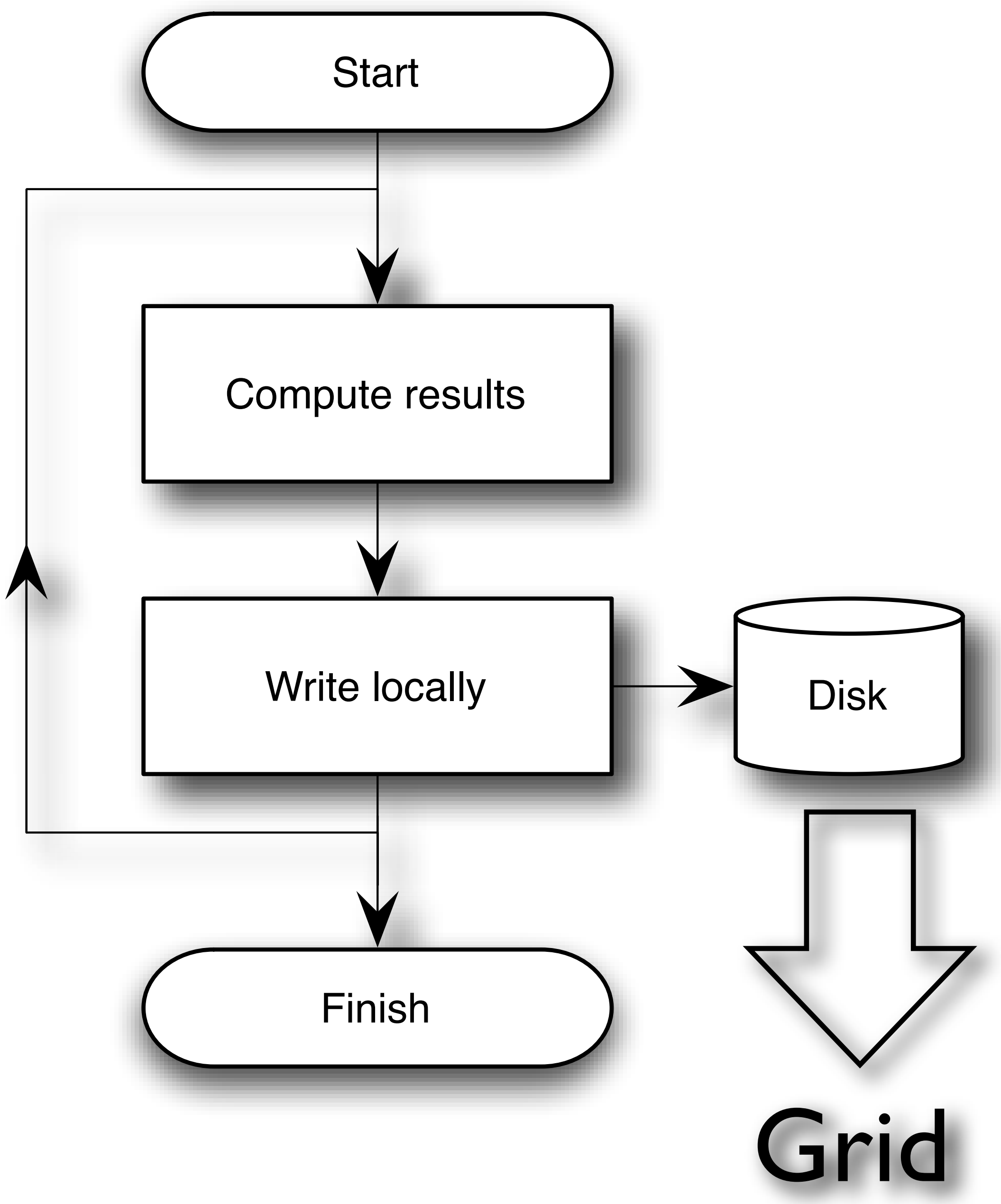
Data flow



Measurable time intervals:

- total turn-over time *T*,
- first turn-over time *F*,
- simulation step time *S*,
- complete simulation time *C*.

Data flow



Measurable time intervals:

- total turn-over time *T*,
- first turn-over time *F*,
- simulation step time *S*,
- complete simulation time *C*.

Reference scenarios:

- serial data last (for minimal *C*),
- serial data first (for minimal *F*).

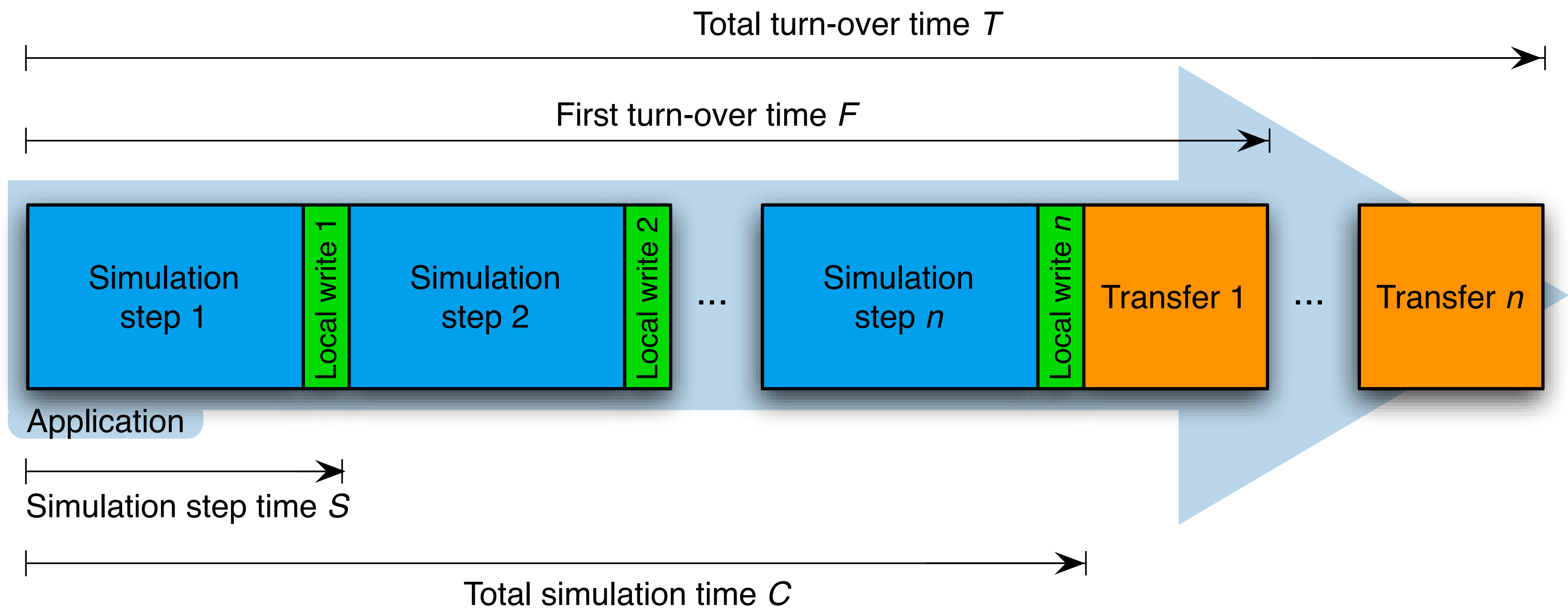
Serial scenarios

Data Last
(minimal *C*)

Data First
(minimal *F*)

Serial scenarios

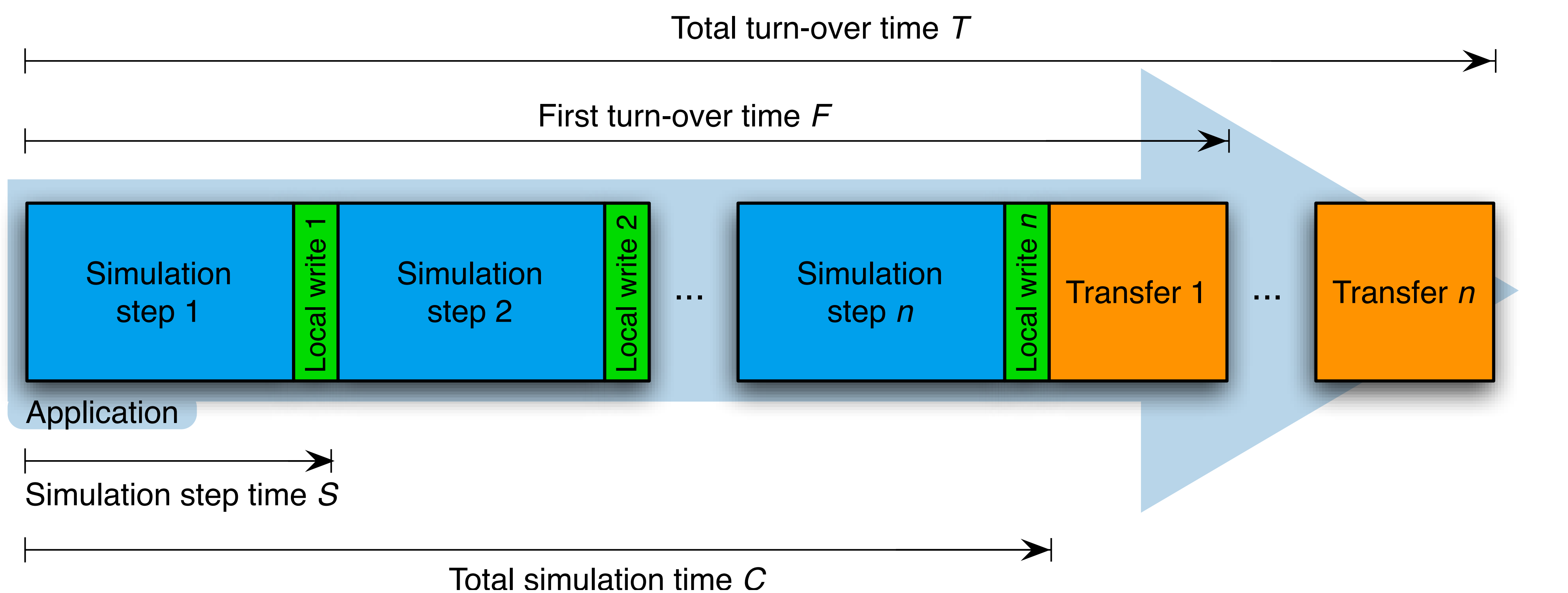
Data Last
(minimal *C*)



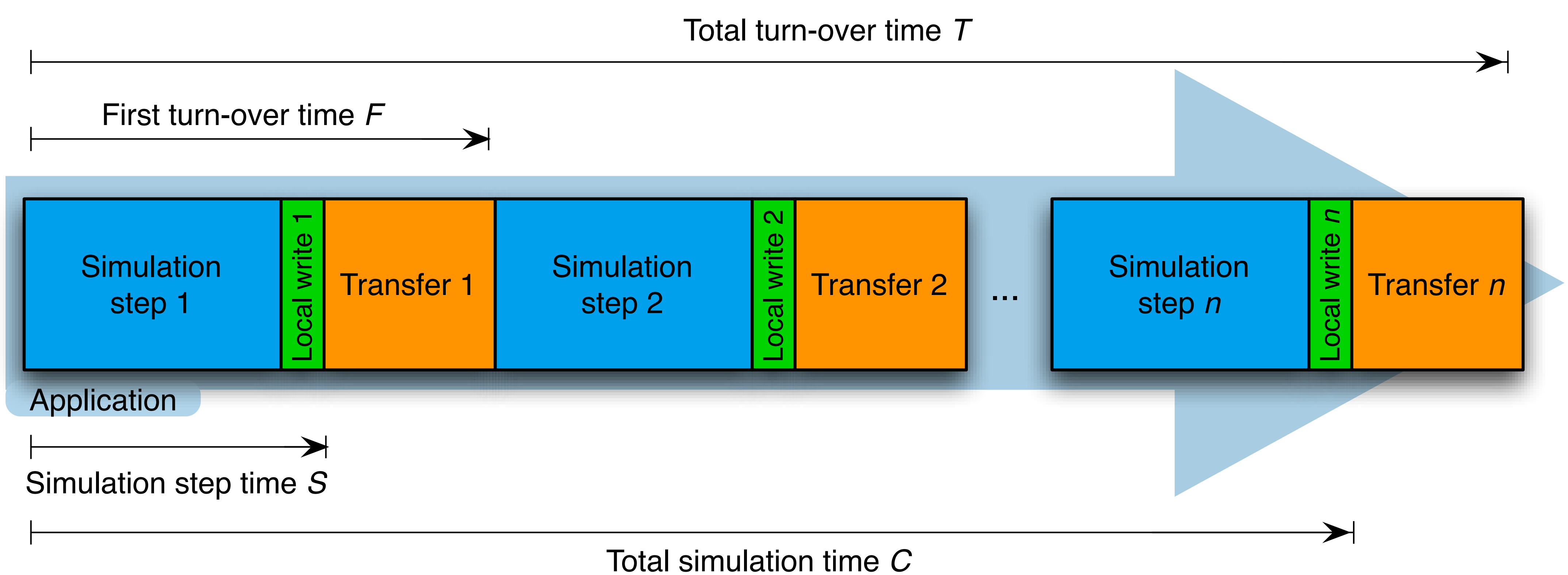
Data First
(minimal *F*)

Serial scenarios

Data Last
(minimal *C*)



Data First
(minimal *F*)



Threaded approach

Grid transfers:

- much slower than disk storage (of course);
- slower than local network transfer (middleware overhead).

Threaded approach

Grid transfers:

- much slower than disk storage (of course);
- slower than local network transfer (middleware overhead).

⇩

Overlap computation with data transfer using threads.

Threaded approach

Grid transfers:

- much slower than disk storage (of course);
- slower than local network transfer (middleware overhead).

⇩

Overlap computation with data transfer using threads.

⇩

Two scenarios to evaluate performance impact: :

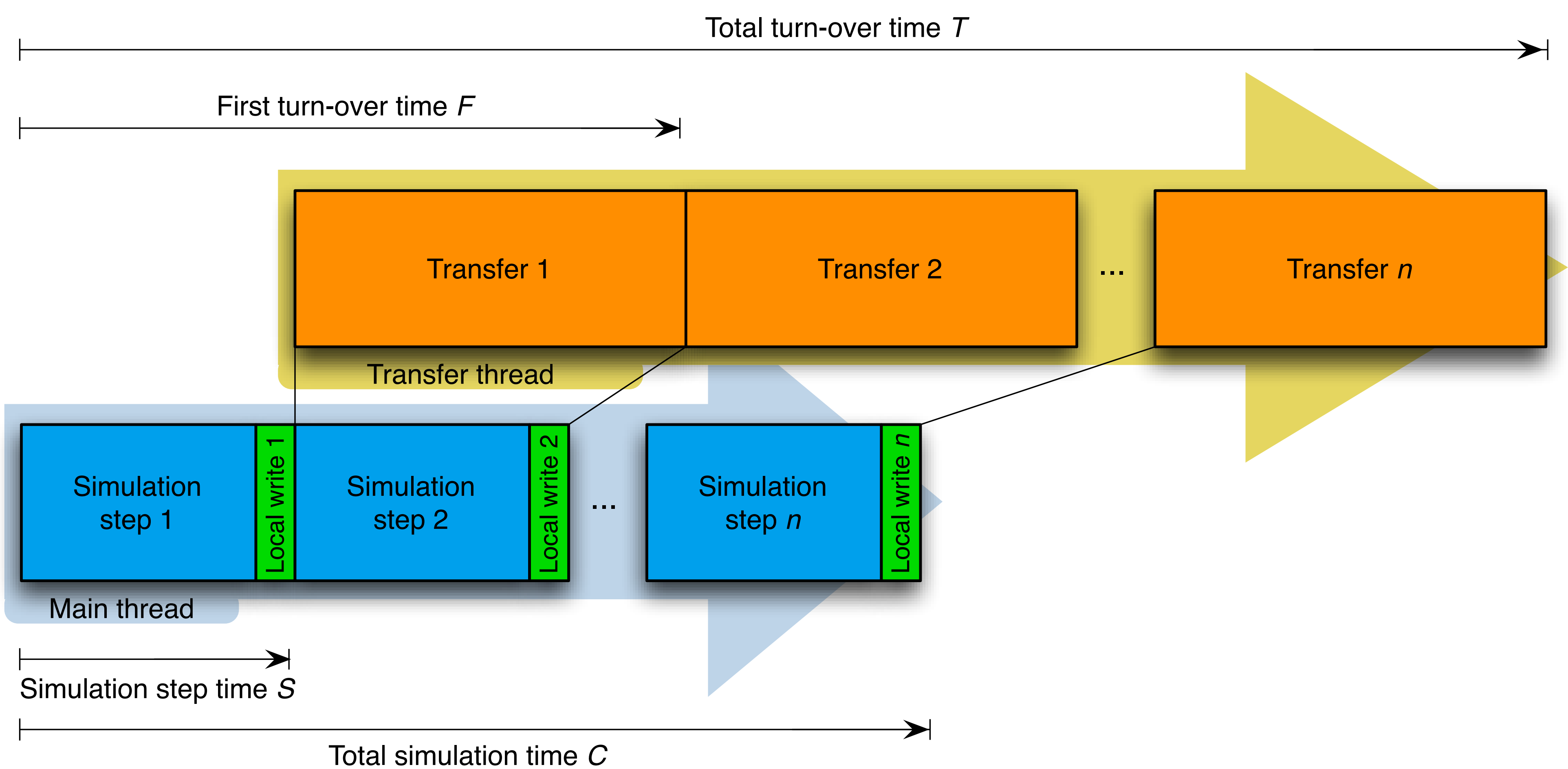
- data **intensive**: transfer takes more time than simulation;
- data **weak**: transfer takes less time than simulation.

Threaded scenarios

Data Intensive
(minimal *T*)

Data Weak
(minimal *T*)

Threaded scenarios



Data Intensive

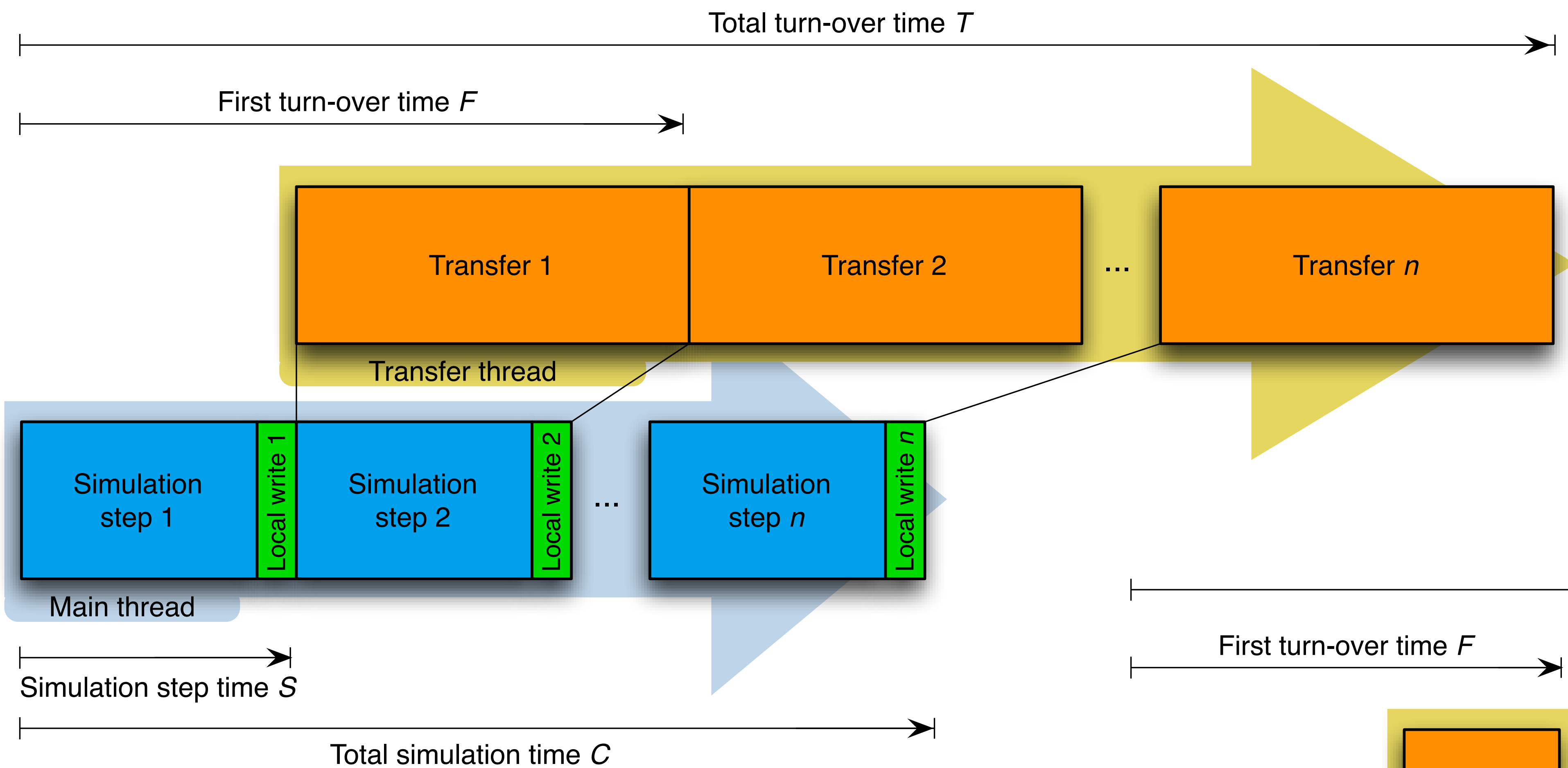
(minimal *T*)

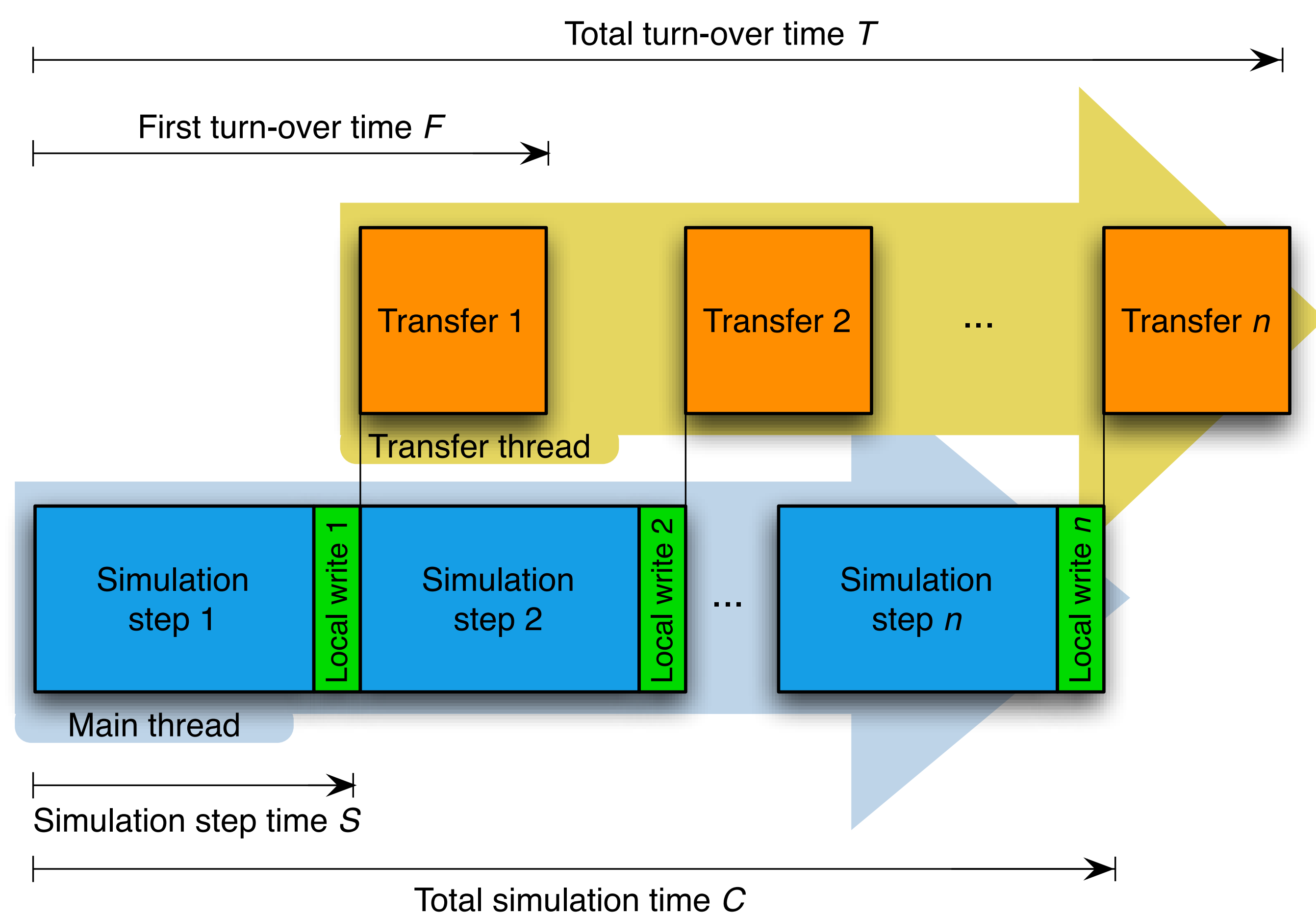
Data Weak

(minimal *T*)

Threaded scenarios

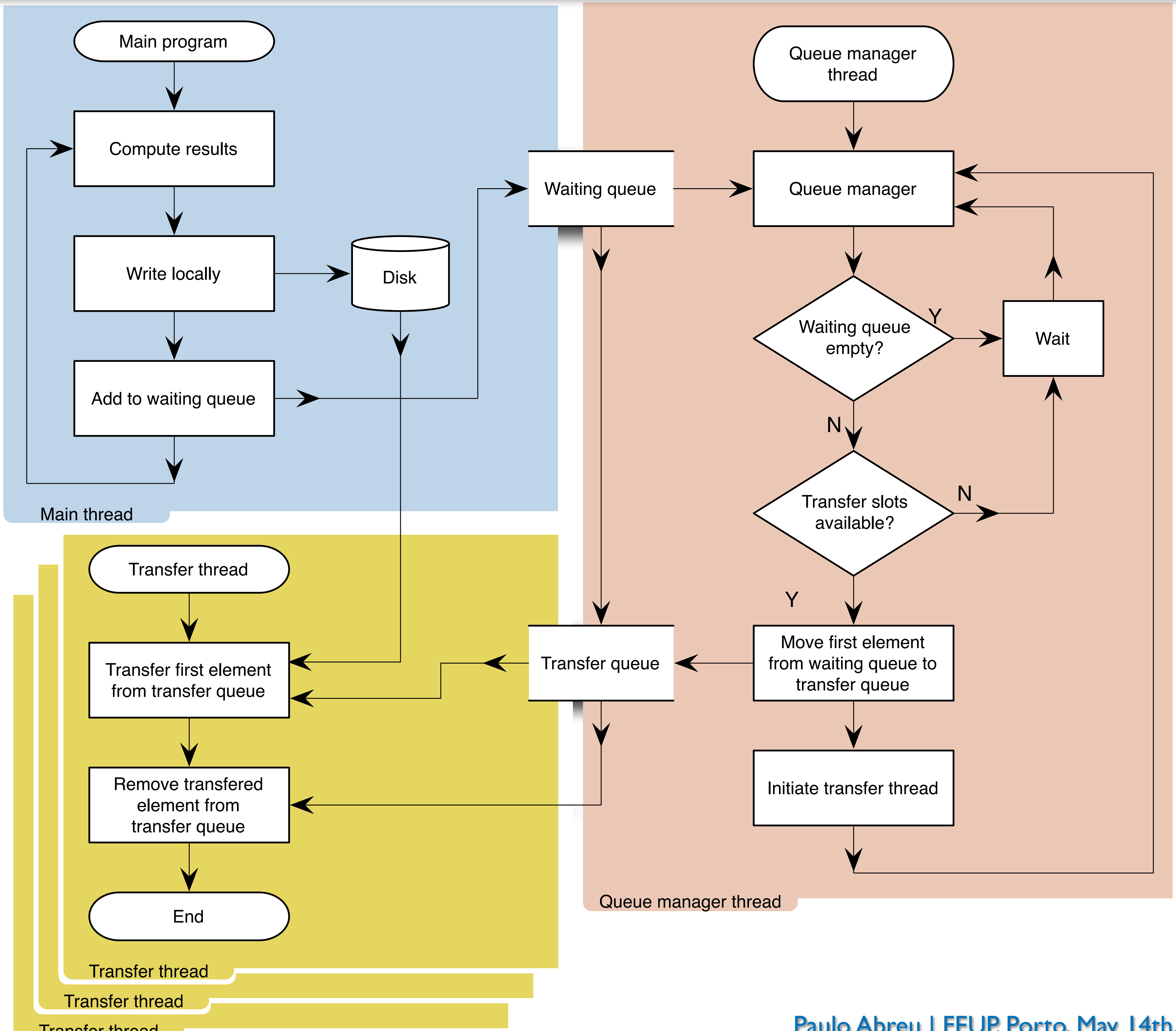
Data Intensive
(minimal T)





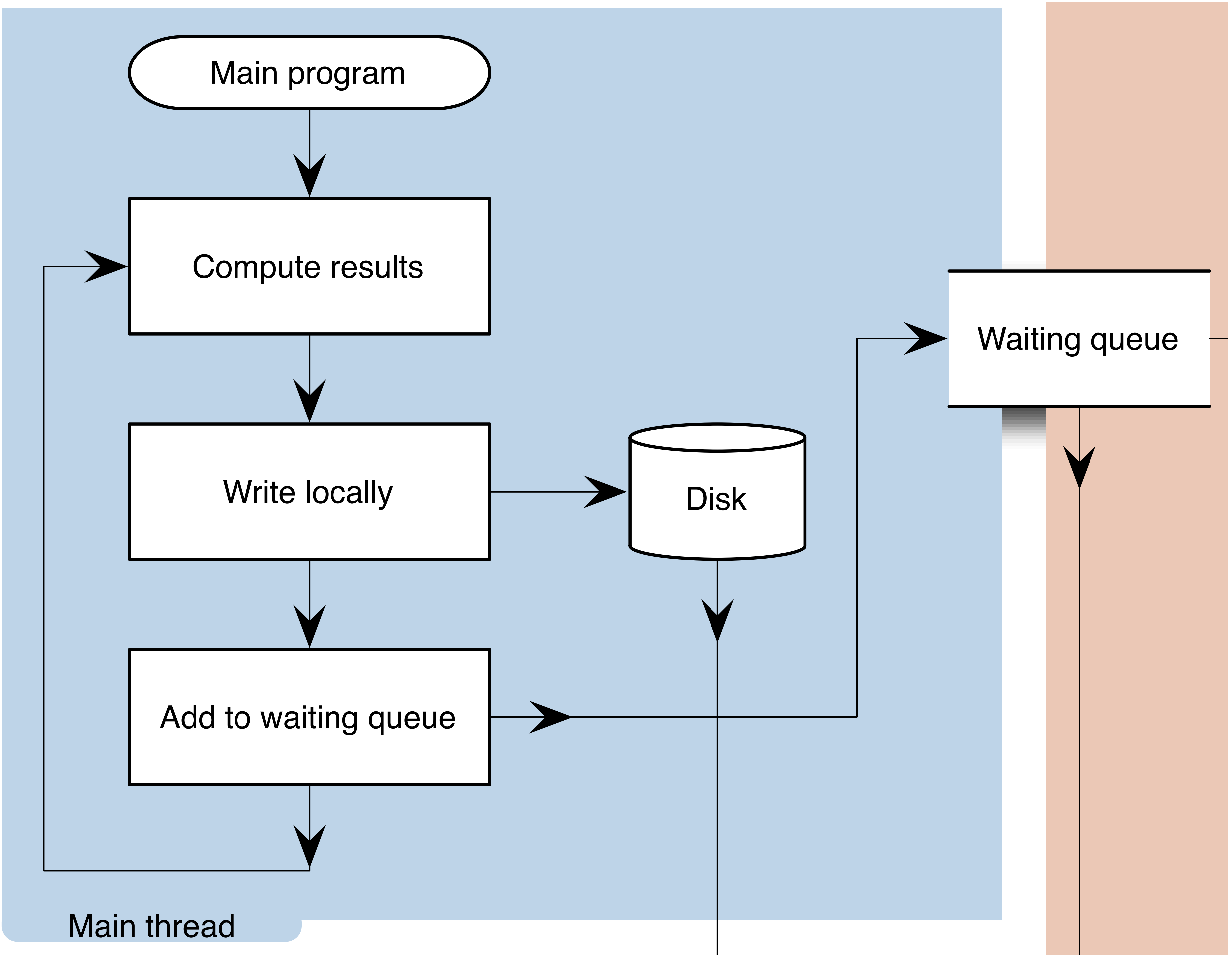
Data Weak
(minimal T)

Algorithm

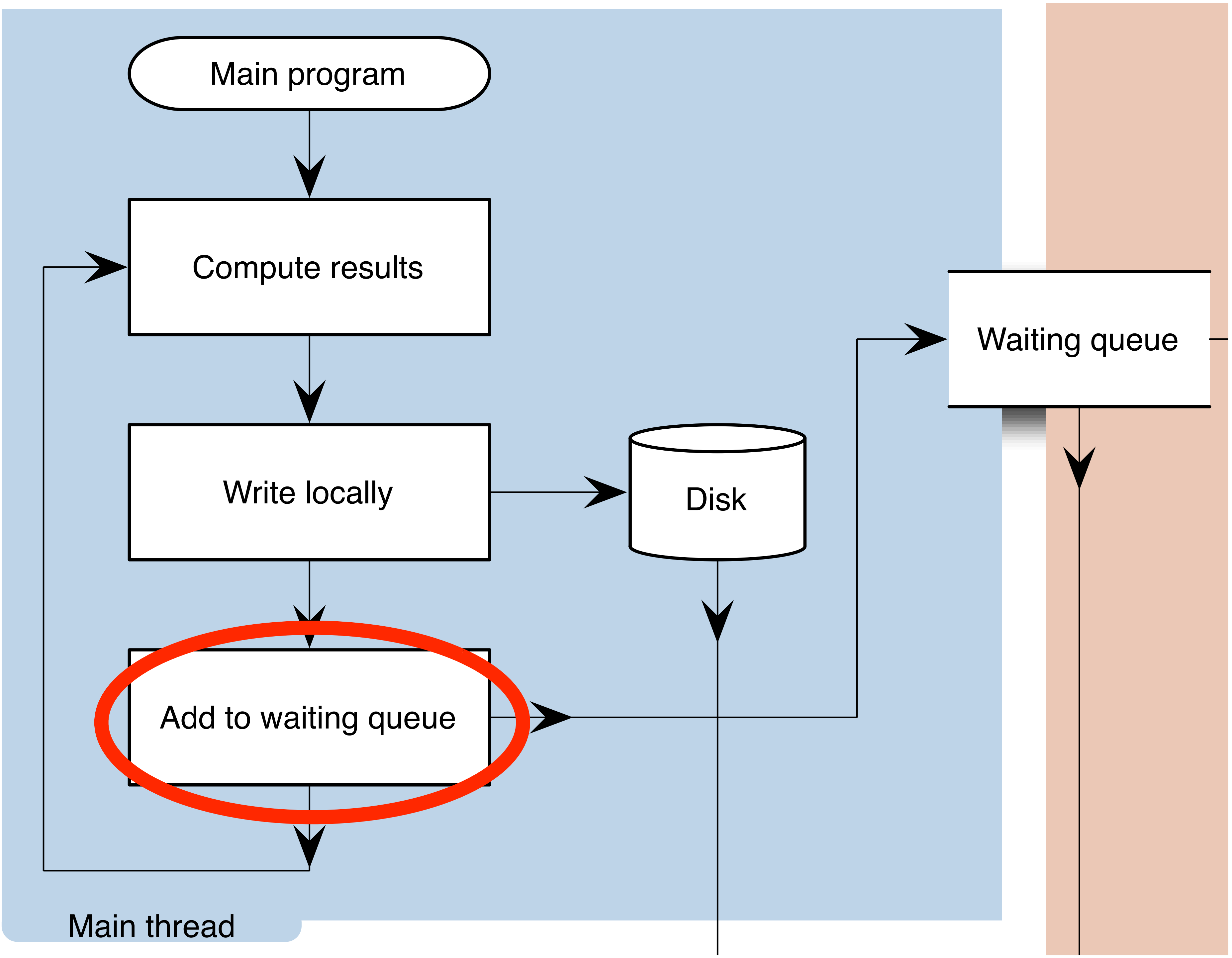


Main program
Compute results
Write locally
Disk
Add to waiting queue
Main thread
Waiting queue
Queue manager thread
Queue manager
Waiting queue empty?
Y
Wait
N
Transfer slots available?
N
Y
Move first element from waiting queue to transfer queue
Initiate transfer thread
Queue manager thread
Transfer queue
Transfer thread
Transfer first element from transfer queue
Remove transfered element from transfer queue
End
Transfer thread
Transfer thread
Transfer thread

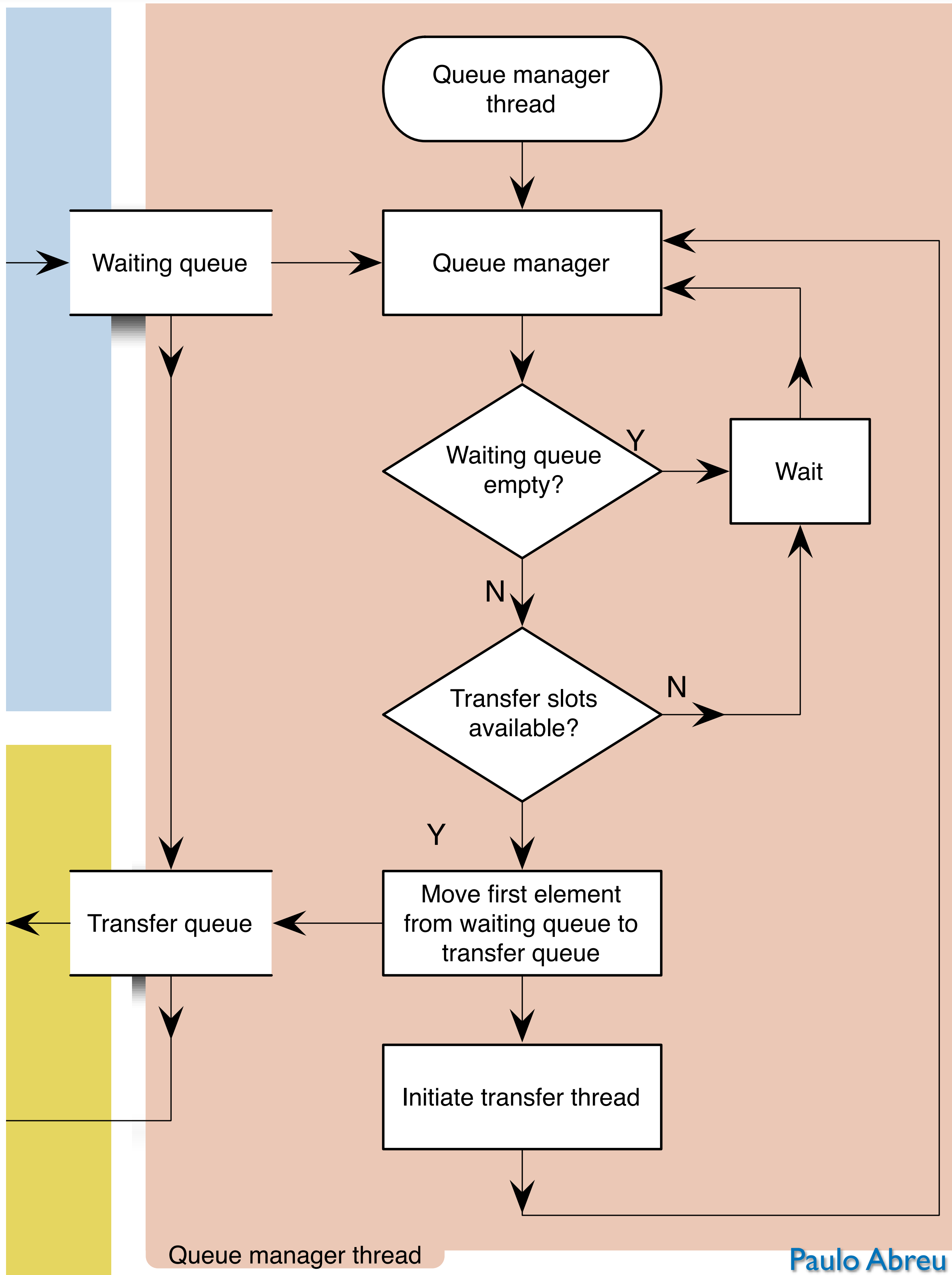
Algorithm: main thread



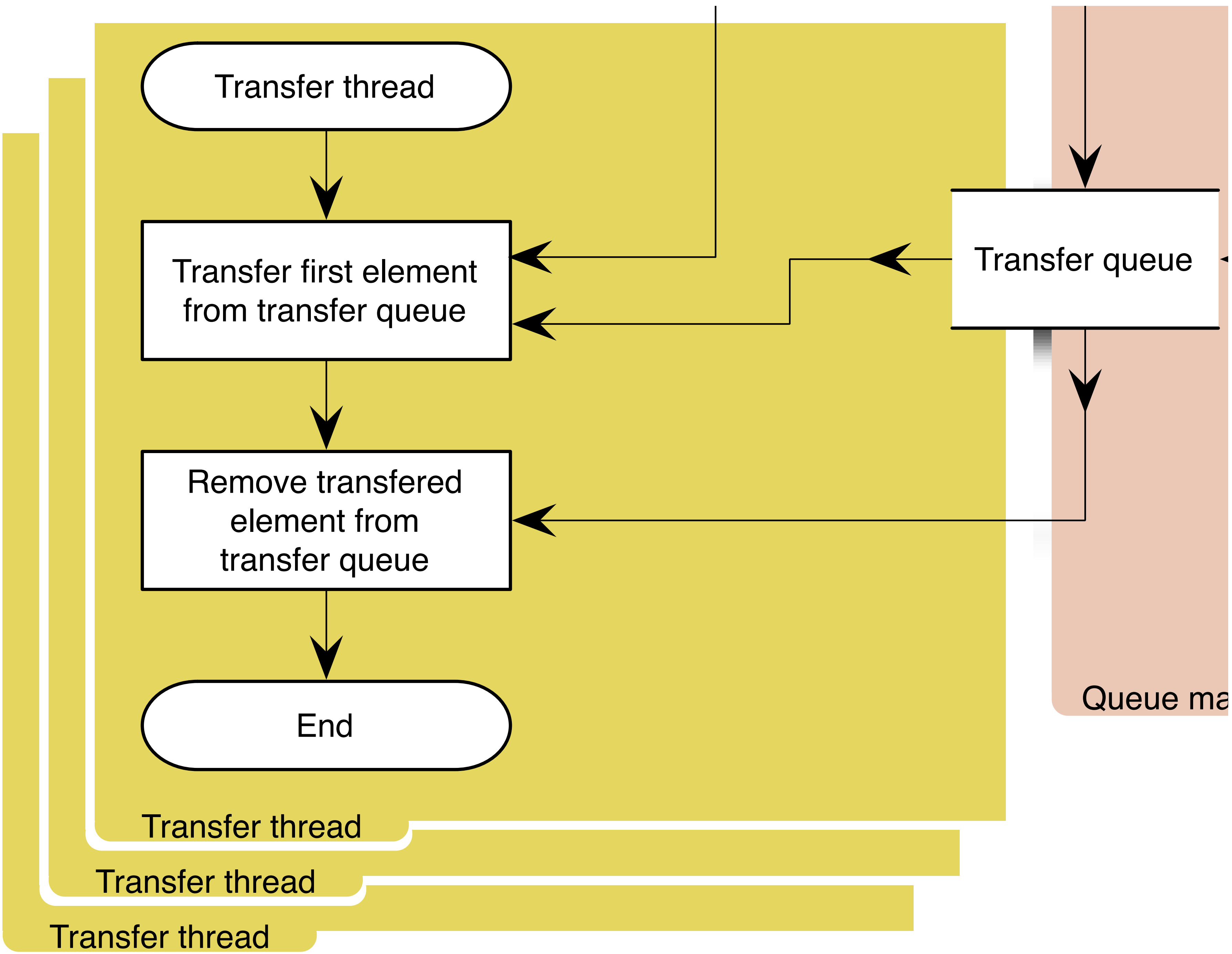
Algorithm: main thread



Algorithm: queue manager thread



Algorithm: transfer threads



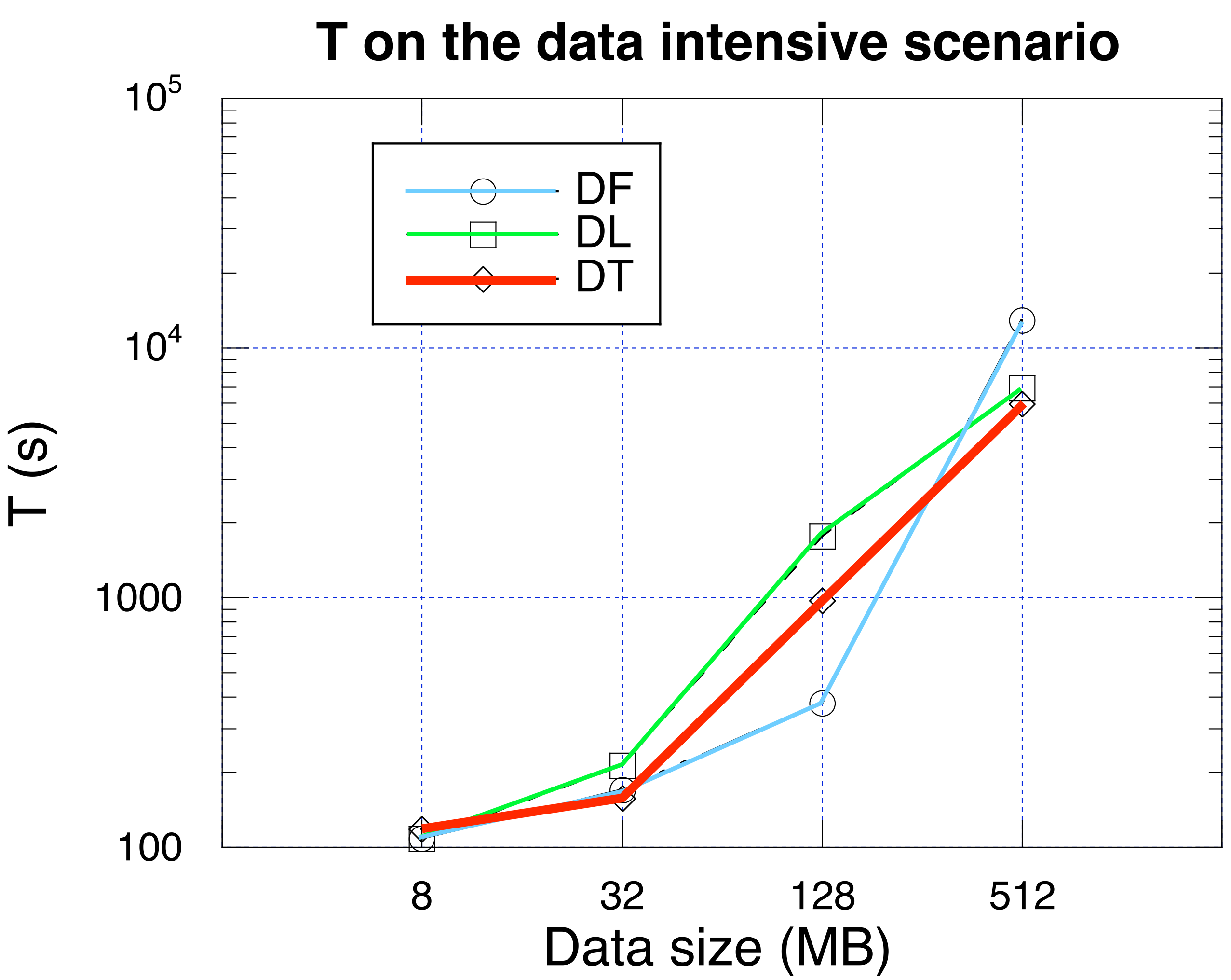
Test application

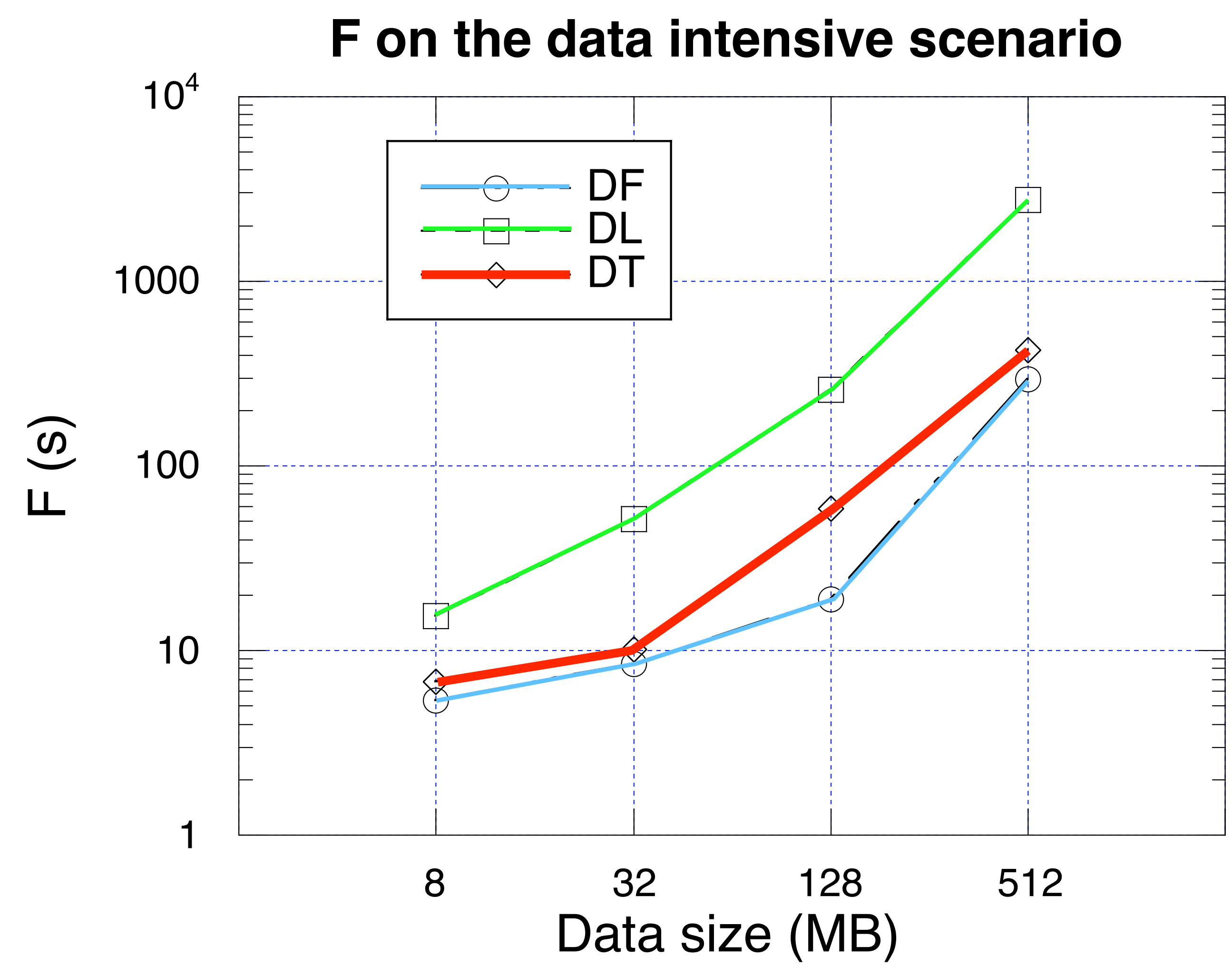
- 20 files of 8MB, 32MB, 128MB and 512MB;
- **timed:** total turn-over time ***T*,** first turn-over time ***F*;**
- evaluated **serial** and **threaded**;
- evaluated **data weak** (calculations >> transfer) and **data intensive** (calculations < transfer).

DW	8MB	32MB	128MB	512MB
S (s)	**27**	**55**	**160**	**487**
Transfer time (s)	**6**	**8**	**13**	**285**

Data intensive results

(calculations < transfer)

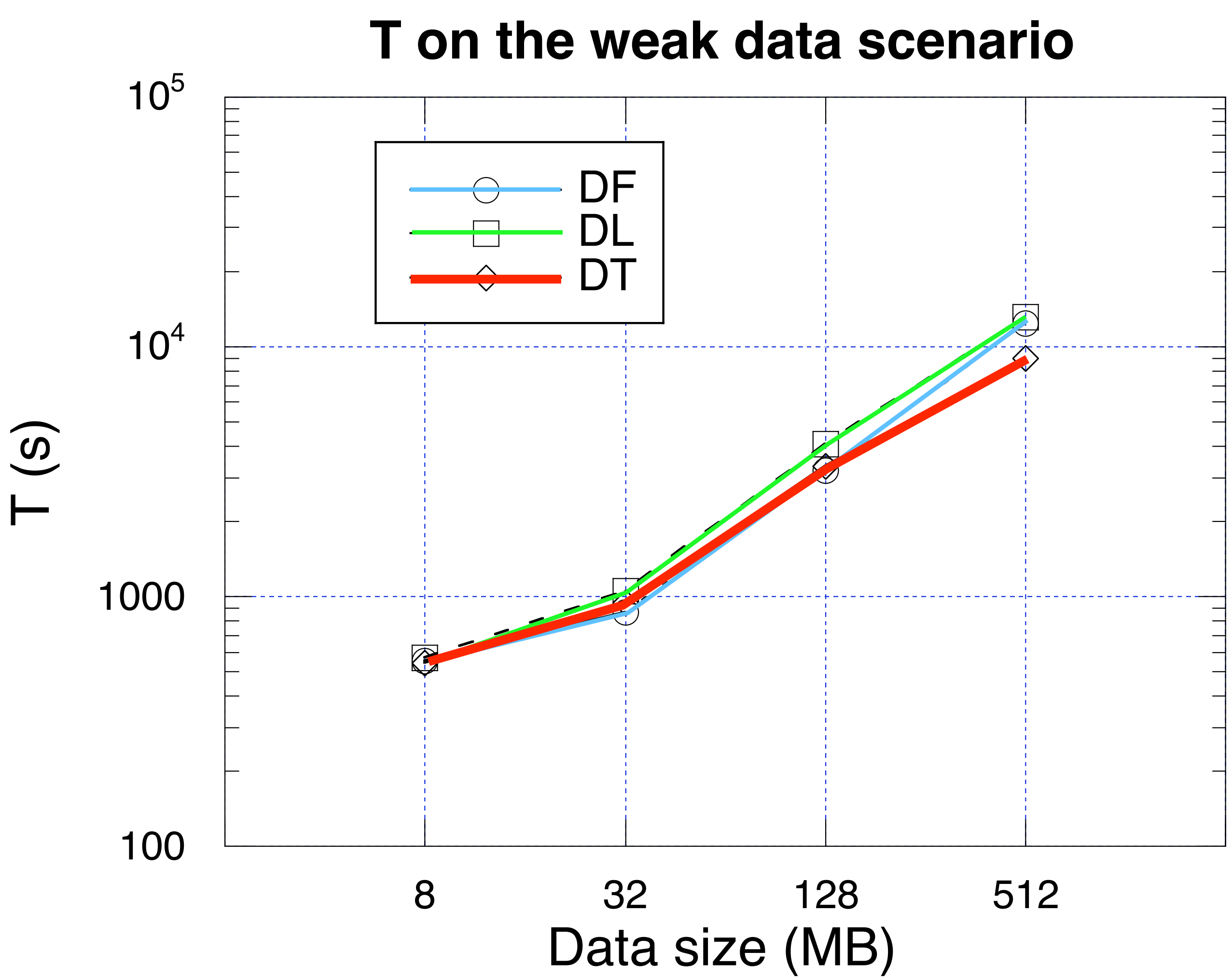


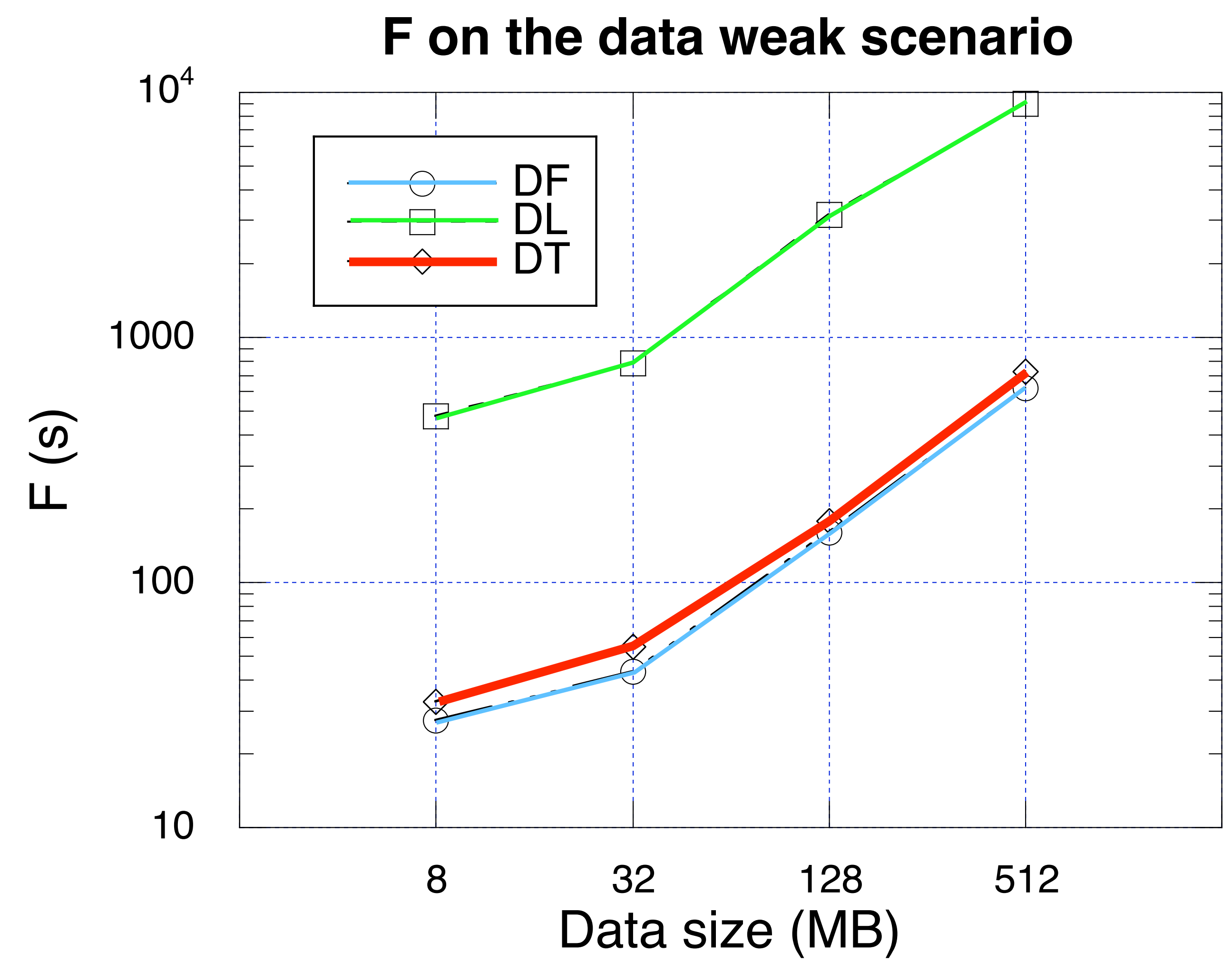


- **Thread performance improves** as the data size increases.
- **Threaded slightly better than serial** for large data files (big variation in 128MB).

Data weak results

(calculations >> transfer)

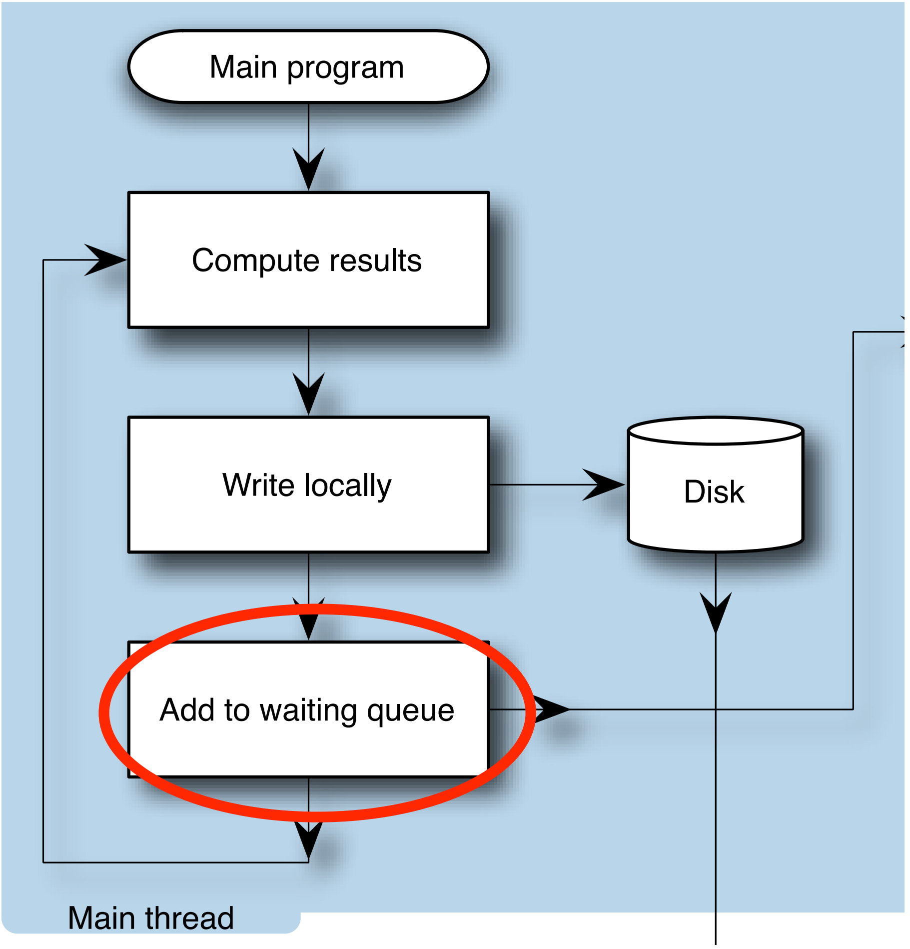




- Threaded performance gain **below expected**.
- **Threaded equivalent to data first** but improves for large data files.

Application deployment

- Simple and efficient API with only two exported C functions:
  - `write_remote(char* fileName):`
    - initializes and adds a file to the transfer queue,
    - called after each file is written to local storage;
  - `write_finished():`
    - finish all transfer threads,
    - called once at the end.
- Fortran 9x wrapper developed.



Application deployment example

Osiris: electromagnetic fully relativistic PIC code for kinetic plasma simulation:

- ~100k lines of Fortran 95,
- HDF5 output,
- complete migration in just a few hours,
- +100 files / hour,
- files sizes from few kB to 100's MB.

Application deployment example

Osiris: electromagnetic fully relativistic PIC code for kinetic plasma simulation:

- ~100k lines of Fortran 95,
- HDF5 output,
- complete migration in just a few hours,
- +100 files / hour,
- files sizes from few kB to 100's MB.

```
    call h5fclose_f(diagFile%id, ierr)

#ifdef __GRID_WRITE_REMOTE__
    ! start grid transfer of file
    call write_remote( trim(diagFile%filepath)
#endif
```

```
   ! close hdf5
   call h5close_f(ierr)

#ifdef __GRID_WRITE_REMOTE__
   ! wait for grid transfers to complete
   call write_finished(ierr)
#endif
```

Application deployment example

Osiris: electromagnetic fully relativistic PIC code for kinetic plasma simulation:

- ~100k lines of Fortran 95,
- HDF5 output,
- complete migration in just a few hours,
- +100 files / hour,
- files sizes from few kB to 100's MB.

```
    call h5fclose_f(diagFile%id, ierr)

#ifdef __GRID_WRITE_REMOTE__
    ! start grid transfer of file
    call write_remote( trim(diagFile%filepath)
#endif
```

```
   ! close hdf5
   call h5close_f(ierr)

#ifdef __GRID_WRITE_REMOTE__
   ! wait for grid transfers to complete
   call write_finished(ierr)
#endif
```

Results:

- desktop/cluster/Grid neutral,
- data available on the SE transparently,
- **improvement of over 35%** from serial to threaded.

	Serial (data first)	Threaded	Difference
T (s)	**3379**	**2164**	**-35,95%**
S (ms)	**386,8**	**447,9**	**+15,80%**

Conclusions

A simple library for threaded transfers:

- files are available to the Grid **as soon as they are produced**;
- **minimal application impact** (to be improved);
- **minimal integration effort**;
- **minimal user knowledge**;
- optimizations underway.

Conclusions

A simple library for threaded transfers:

- files are available to the Grid **as soon as they are produced**;
- **minimal application impact** (to be improved);
- **minimal integration effort**;
- **minimal user knowledge**;
- optimizations underway.

Next steps:

- more performance tests with other parameters (file size, frequency, ...);
- allow for several transfer threads;
- set SE at run time;
- add initialization function;
- explore alternatives.

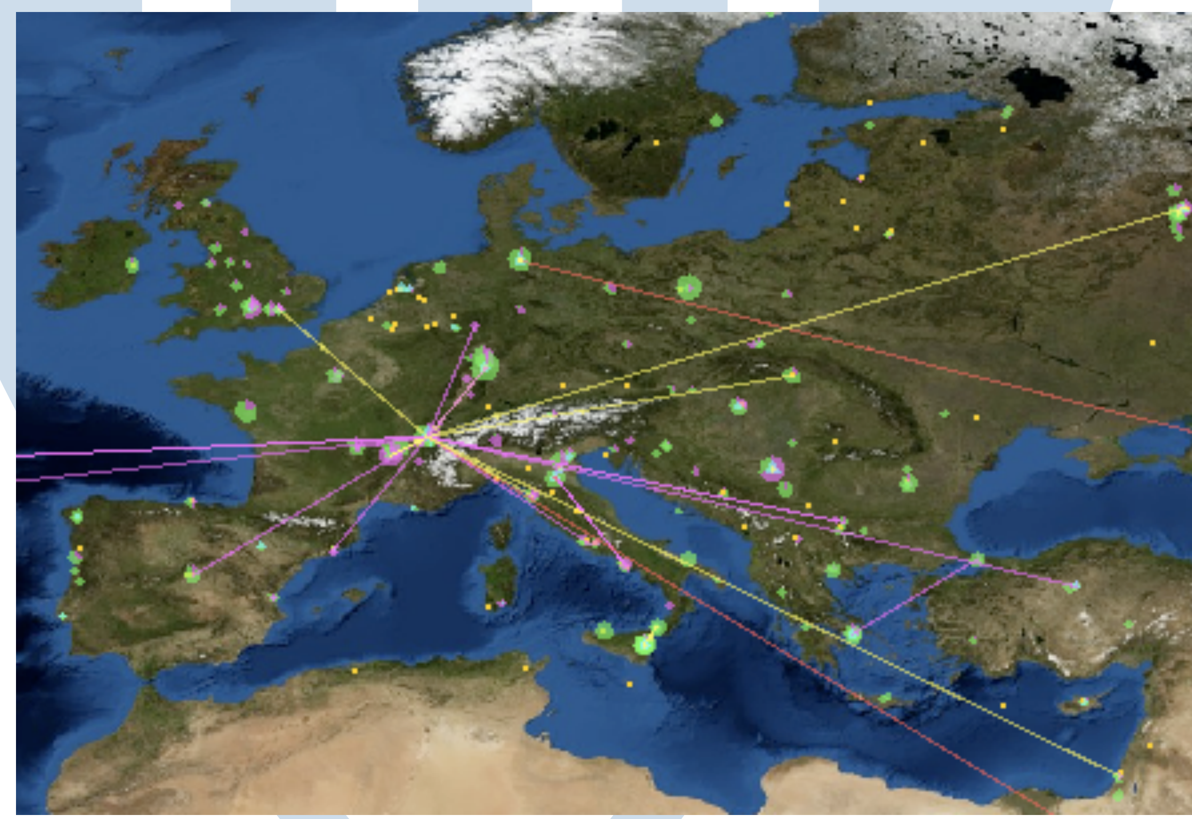
Migrating large output applications to Grid environments:

a simple library for threaded transfers with gLite

Paulo Abreu

Ricardo Fonseca
(GoLP/IST, DCTI/ISCTE)

Luís O. Silva
(GoLP/IST)



GoLP/Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico (IST)
Lisbon, Portugal
http://cfp.ist.utl.pt/golp