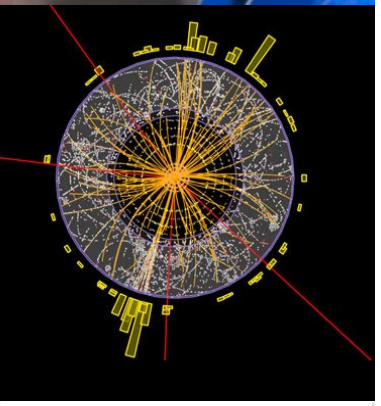
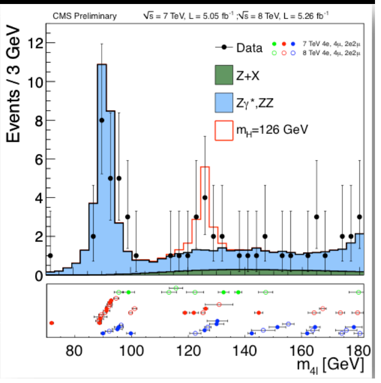
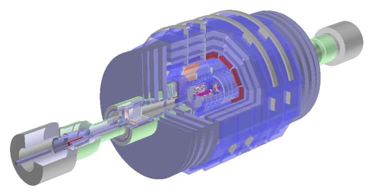
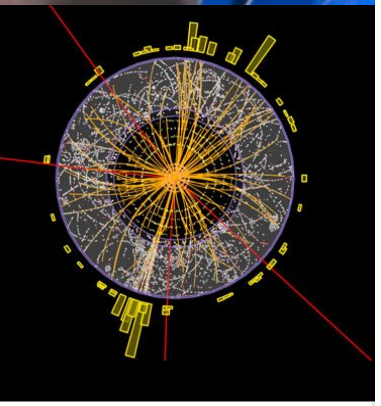
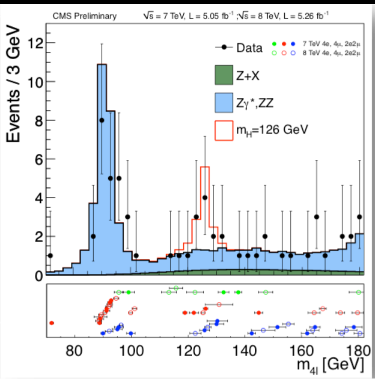



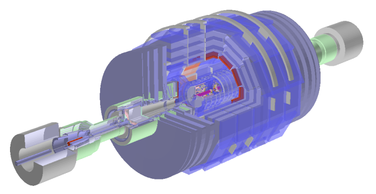


ROOT I/O Review and Future Plans

Philippe Canal
Fermilab

Overview



- The long road to Root 6
  - What's in a name
  - What's next
  - Testing
- Other Trends
  - Parallel Processing
  - Optimization
  - File Format Upgrades
  - I/O Customization Framework
  - ***TTree***
- Challenges and outlook

The Long I/O Road to ROOT 6

- rootcling
  - Migrate from ***CINT*** interface to ***Clang*** AST
  - Implement new LinkDef parser and new selection mechanism
  - Deal with the different naming conventions for C++ entities
  - **Add support for (quasi) opaque typedef**
    - Also in default template parameters!
  - Replace ShowMembers
    - Switch from generated code to just-in-time analysis of AST.
  - Migrate access to Class annotation and docs strings.
  - Migrate type search (lookup) routines
- Core/Meta
  - See Cling presentation
  - Deal with the different naming conventions for C++ entities

What's in a name …

- ***CINT*** and ***C++*** names are quite different
  - Implicit using namespace std statement in ***CINT***.
  - User typed spelling vs. 'real' spelling
    - ***vector<Int_t>*** vs ***std::vector<int, std::allocator<int> >***
    - User typed spelling not always available in Clang, especially for derived entities (data member of templates).
  - ***Clang*** does not propagate typedef to default template args
  - ***CINT*** template parsing bugs/shortcuts.
  - Opaque typedefs (***Double32_t***, ***std::string***, etc…)
- Clang and gcc(xml) names are similar

- Almost sole source of 'risk' left for ***I/O***

What's in a name

- Implemented normalization routines that
  - Adds full qualification
  - Adds default template parameter except for ***STL*** containers
  - Keeps opaque typedefs
- Extra care to preserve user typed spelling and be as close as possible to the "***ROOT I/O*** name"
- ***However*** some names ***must*** change
  - ***Outer::Tplt<Inner>*** `->` ***Outer::Tplt<Outer::Inner>***
  - Adding missing default template arguments
- Risk/Consequences alleviated by
  - Renaming I/O customization rules
  - Automatic matching of different spelling
  - Added flexibility in checksum matching cross-checks

Backward Incompatibility

- Name changes (as just mentioned)

- ***rootcling*** no longer re-#defines the private and protected keywords to public.
  - *ACLiC* no longer breaks privacy!

- As a consequence I/O is ***currently*** not supported for private or protected classes
  - The major issue is access the constructor and destructor

ROOT 6 I/O Testing

- Backward and Forward compatibility testing

- ***roottest*** and sets of known files used to check v5 read in v6 and vice et versa
  - Leverage ***MakeProject*** to 'rewrite' some files in v6.

- Once v6 beta is available to the experiments
  - a part of validation must be to try reading v5 files in v6
  - and vice et versa

What's next …

- Making sure "class renaming" support in I/O customization framework works in all necessary cases

- ***Genreflex*** command line
- ***Selection.xml*** parsing
  - Real life (standalone☺) examples welcome from experiments

- More verifications on opaque typedef and templates
  - Risk reduced thanks to automatic conversions

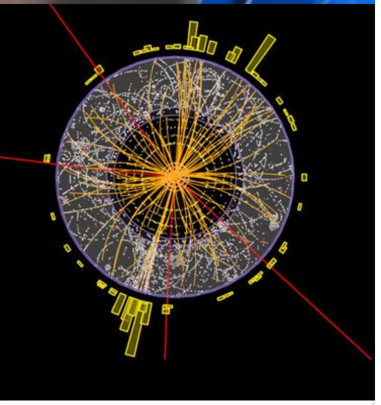
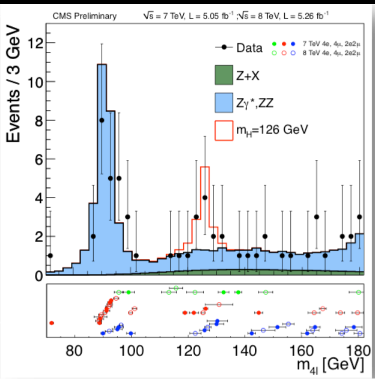
- Implement support for ***I/O*** for private classes

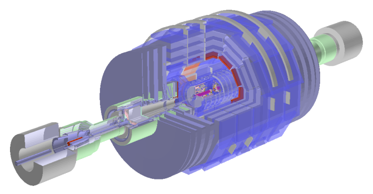
Here comes cling

- ***Cling*** introduces binary compatible Just In Time compilation of script and code snippets.

- Will allow:
  - ***I/O*** for 'interpreted' classes
  - Runtime generation of ***CollectionProxy***
    - Dictionary ***no longer*** needed for collections! ***[Summer Student]***
  - Run-time compilation of ***I/O*** Customization rules
    - including those carried in ***ROOT*** file.
  - Derivation of 'interpreted' class from compiled class
    - In particular ***TObject***
  - Faster, smarter ***TTreeFormula***
  - Potential performance enhancement of ***I/O***
    - Optimize hotspot by generating/compiling new code on demand
  - Interface simplification thanks to full ***C++*** support
    - New, simpler TTree interface (***TTreeReader***) ***[Summer Contributor]***

Other Trends



- Parallel Processing
- Optimization
- File Format Upgrades
- I/O Customization framework
- ***TTree***

Contributors

- Philippe: 25% on I/O
- Punctual effort from experiment (ROOT I/O Workshop)

Thread Safety

- ***Cling*** enables support for robust multi-thread ***I/O***
  - ***Cling*** has clear separation of database engine and execution engine allowing to lock them independently

- Currently multi-threaded ***I/O*** supported as long as
  - All the ***TClass*** and ***TStreamerInfo*** are (explicitly) created serially.
  - Each ***TFile*** and ***TTree*** objects are accessed by only one thread (or the user code is explicitly locking the access to them).

- ***Cling*** will allow to remove the first limitation.

Why one thread/schedule per TTree

- When reading TTree holds:
  - Static State:
    - List of branches, their types their data location on file.
  - Dynamic State:
    - Current entry number, ***TTreeCache*** buffer (per ***TTree***), User object ptr (one per (top level) branch), Decompressed basket (one per branch)
  - Separating both would decrease efficiency
- Advantages
  - Works now!
  - No need for locks or synchronization
  - Decoupling of the access patterns
- Disadvantages
  - Duplication of some data and some buffers.
    - However this is usually small compare to the dynamic state.
  - Duplication of work if access overlap

Parallel Merge

- New class ***TMemFile***
  - A completely in memory version of ***TFile***
  - Support one thread/schedule per TTree pattern without costing disk I/O time
- New class ***TParallelMergingFile***
  - A ***TMemFile*** that on a call to Write will
    - Upload its current content to a parallelMergerServer
    - Reset the ***TTree*** objects to facilitate the new merge.

```
TFile::Open("mergedClient.root?pmerge=localhost:1095","RECREATE");
```

- New daemon parallelMergeServer
  - Receive input from local or remote client and merges into requested file (which can be local or remote)
  - Fast merge ***TTree***. Re-merge all histogram at regular interval

Parallel Merge Challenges

- Efficiently deal with many histograms
  - Each of them still need to be merged at the end
- Lack of ordering of the output of the workers
  - No enforcing of luminosity block boundaries for example
  - Introducing support for the ordering would lead to increased interdependency between the worker and the server
  - Advanced space reservation is challenging due to the variable size of the entries.

Parallel merge

- Time scale for a fully tested and performing version.
  - 'multi-process' version around 6 months
    - Parallel Merge Daemon (authorization, auto-start, error handling)
    - ***Parallel Merge for Histogram*** (proper set of benchmarks, performance improvement, etc.)
  - 'multi-thread' version requires v6 and additional thread safety fixes.

- Benchmarks
  - Still to be designed
  - Based on existing example (some multithread) and new example based of the ***Event*** test.
  - Based on experiment uses cases.

Other Possible Parallel Processing

- Read/Write branches using *internals* thread/tasks
  - Need to partially back out memory optimization
  - Require ***TFile*** synchronization
- Read/Write branches in multiple user thread
  - Need to design the limit and semantics
  - Extra complexity to conserve basket clustering
  - Require ***TFile*** synchronization
- Offload work (compression) to separate thread
  - Need to work well with task based scheduler
- Thread safe version of ***TFile***
  - Not quite sure of semantic
  - Need to be cost-neutral for traditional uses.

Optimizations

- ***OptimizeBasket***
  - There are a couple of new algorithm proposals
  - Need to be tested on wide range of cases
- Read/WriteBuffer
  - 25% of the read code moved to optimized framework (function based) ; representing most of the use cases.
  - Write code still need to be similarly optimized
- ***TTreeCache***
  - Start using it in ***TTreeCloner***.
  - Allow alternative algorithm
  - Tests, tests and tests
  - Switch on by default.

File Format Upgrades

- Write-once files
  - Support for direct write to ***Hadoop*** file System
- Switch to little-endian
  - Enable additional run-time optimization
- Improve self-description meta-data
  - Store also typedef information.
- ***SQLite*** within ***ROOT*** file
  - Support database (for meta-data) co-located with data
- Space saving changes.
  - Improve compression of branch of unsplit collections
  - Reduce overhead for deep hierarchy
- Time saving changes
  - Compress each entry individually to improve random access

I/O Customization Framework

- Bug fixes
  - Class renaming
  - Rules execution in complex ***TTree***
- Continue development
  - Extend documentation
  - Implement Write rules
  - Enable Just-In-Time compilation of rules
- Extend automatic conversions
  - ***Derived**** <-> ***Base****
  - From object to pointer

TTree

- ***TTree***
  - Bug fixes
  - Interface simplification
    - ***TTreeReader*** (as clamored for in workshop) ***[External Contribution]***
    - Make ***SetAddress*** and ***SetBranchAddress*** ‘smarter’
  - Optimizations
  - Improve documentation
  - Improve statistics gathering ***[Atlas]***
- ***TTree*** Draw/Scan
  - Add support for 64bit integer calculation ***[Atlas]***
  - Leverage cling

Challenges

- Large program of work
  - 42 outstanding deficiencies
  - 62 improvements and new features
- Effort
  - My effort spread over ***ROOT I/O***, ***Cling*** and ***Geant/GPU***
    - Split 50/50 between ROOT and Geant
  - Extra effort required to make any real progress
    - Next slides assumes extra effort (.5 FTE)
  - ***ROOT I/O*** Workshop helps coordinate direct effort from experiments
    - This comes and goes 'as needed' and competes with their own internal efforts.
  - Summer Students and other external contribution
    - ***TTreeReader***
    - Runtime generation of ***CollectionProxy***

Priorities Recapitulations – Nov Rel.

- Fix blocking issues / User Support

***Red items** only possible with extra effort.* !

- Required for ROOT 6 beta release
  - Renaming rules - 2w – ***July*** (5035,3211,3670,3708,5264)
  - Genreflex – ***August*** (see cling)
- Multi Processing
  - *First new revision on histogram parallel merge - 3w* – ***September*** (5071)
  - *Parallel merge daemon* – 2w – ***October*** (5070)
- File Format upgrades
  - *Write only once files (Hadoop)* – 1w - ***September*** (5075)
  - *Switch from big endian to little endian* – 1w - ***October*** (5073)

Priorities Recapitulations – Nov Rel.

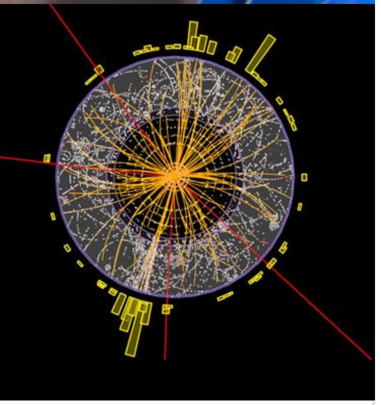
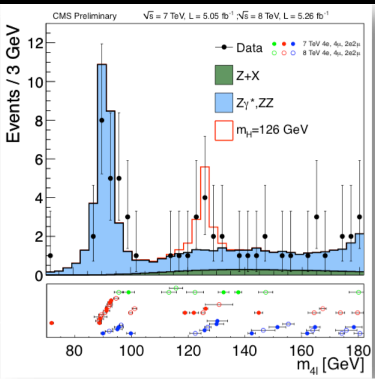
- Performance
  - TTreeCache and TTreeCloner – 1w – ***August*** (5078)
  - *Testing plan for OptimizeBasket,TTreeCache* – 2w- ***September*** (5080)

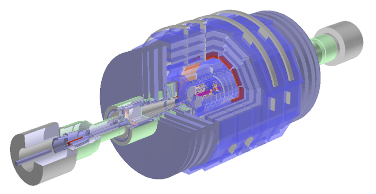
- New Features
  - TTreeFormula and long long ***[Atlas]*** (5084, 5085)
  - TTreePerfStat and multiple TTree ***[Atlas]*** (5079)

- Nice to have
  - TTreeReader ***[External Contribution]*** (5165)
  - Runtime generation of CollectionProxy ***[Summer student]*** (5164)

Priorities Recapitulations – May Rel.

- Fix blocking issues / User Support
- More documentations and fix more outstanding issues.
  - See detailed list …
- Multi-processing
  - Refine parallel merging based on user experience
  - Start upgrading to support multi-threading/tasking
- File Format upgrades
  - Cost of repeated [deep] hierarchies
  - Write I/O customization Rules
- Performance Improvements
  - OptimizeBasket
- Interface Simplification
  - SetBranchAddress, TTree::Draw, etc.



Backup slides

1 year outlook

End Of		Philippe Only		Philippe and extra effort
			3798	*The various TTree::Branch functions are very hard to figure out*
			3992	*TSelector::Process() on TChain*
			5078	Update fast-merging to leverage the TTreeCache
			4549	TRefArray does not clean fUIDs array in Streamer
	4489	Memory leak when TTree::BuildIndex is called multiple times	4550	TMessage doesn't honour kIsOwner bit when compression is used
July	4549	TRefArray does not clean fUIDs array in Streamer	4489	Memory leak when TTree::BuildIndex is called multiple times
			5070	Parallel merging daemon
			4044	*Documentation of compress parameter of TFile::Open()*
				Genreflex replacement
August		Genreflex replacement	**5080**	**Develop a comprehensive test plan for OptimizeBasket, LearnPrefill, TTreeCache.**
			5079	*Update TTreePerfStats to support multiple cache per file (Peter)*
	5079	*Update TTreePerfStats to support multiple cache per file (Peter)*	*5085*	*TTreeIndex supporting Long64_t (Peter)*
	5085	*TTreeIndex supporting Long64_t (Peter)*	*5084*	*TTreeFormula calculation in Long64_t (Peter)*
	5084	*TTreeFormula calculation in Long64_t (Peter)*	**5071**	**Parallel merge of histograms**
September	114	Fix issues in the renaming of classes in split branches where it is the base classes	5075	Write only once files (Hadoop)
			4496	*TTree doc*
			5073	Explore changing the on-file byte format to little endian!
October	*5078*	*Update fast-merging to leverage the TTreeCache*	*4441*	*hadd crashes when merging ntuples with different formats*
		Release Cut off		
			114	Fix issues in the renaming of classes in split branches where it is the base classes
November	5070	Parallel merging daemon	4839	TTree::Refresh and TTree::GetEntry causing crash
			113	*Fix issues when the target of the rule is an 'unsigned int' and when it is a struct*
			3709	*Crash when writing object with schema rule*
December	*5073*	*Explore changing the on-file byte format to little endian!*	*5157*	*Enhance Documentation for I/O customization rules*
			5077	*Find a way to avoid storing the byte count and version number for deep hierarchy!*
	113	Fix issues when the target of the rule is an 'unsigned int' and when it is a struct	*5082*	*Upgrade SetAddress and SetBranchAddress!*
January	3709	Crash when writing object with schema rule	*131*	*Optimize Baskets*
			3078	*Schema evolution rules not applied when loading from TTree*
			4049	**Base class schema problem when using member wise streaming**
			5156	*TTree::Draw and existing histogram*
February	*5075*	*Write only once files (Hadoop)*	*5183*	*TTree c'tor should take TDirectory*
	4550	*TMessage doesn't honour kIsOwner bit when compression is used*	*5066*	*multi-threaded file compression (tree writing)*
March	4833	TMessage::ReadObjectAny returns non-null pointer even in case of errors	*4441*	*hadd crashes when merging ntuples with different formats*
			4444	*ROOT crashes reading bad.root file (II)*
			4576	*Error reading older version ROOT tree file after upgrading ROOT*
April	**4049**	**Base class schema problem when using member wise streaming**	*119*	*Implement Write rules*
		Release Cut off		
	4839	*TTree::Refresh and TTree::GetEntry causing crash*	*5076*	*In TBasket compress each entry individually (for large basket)!*
May	5173	Issue with collection proxy and emulated class	*5159*	*Improve TTree documentation about SetMakeClass()*

1 year outlook

- … Not counting unexpected but essential new issues ….

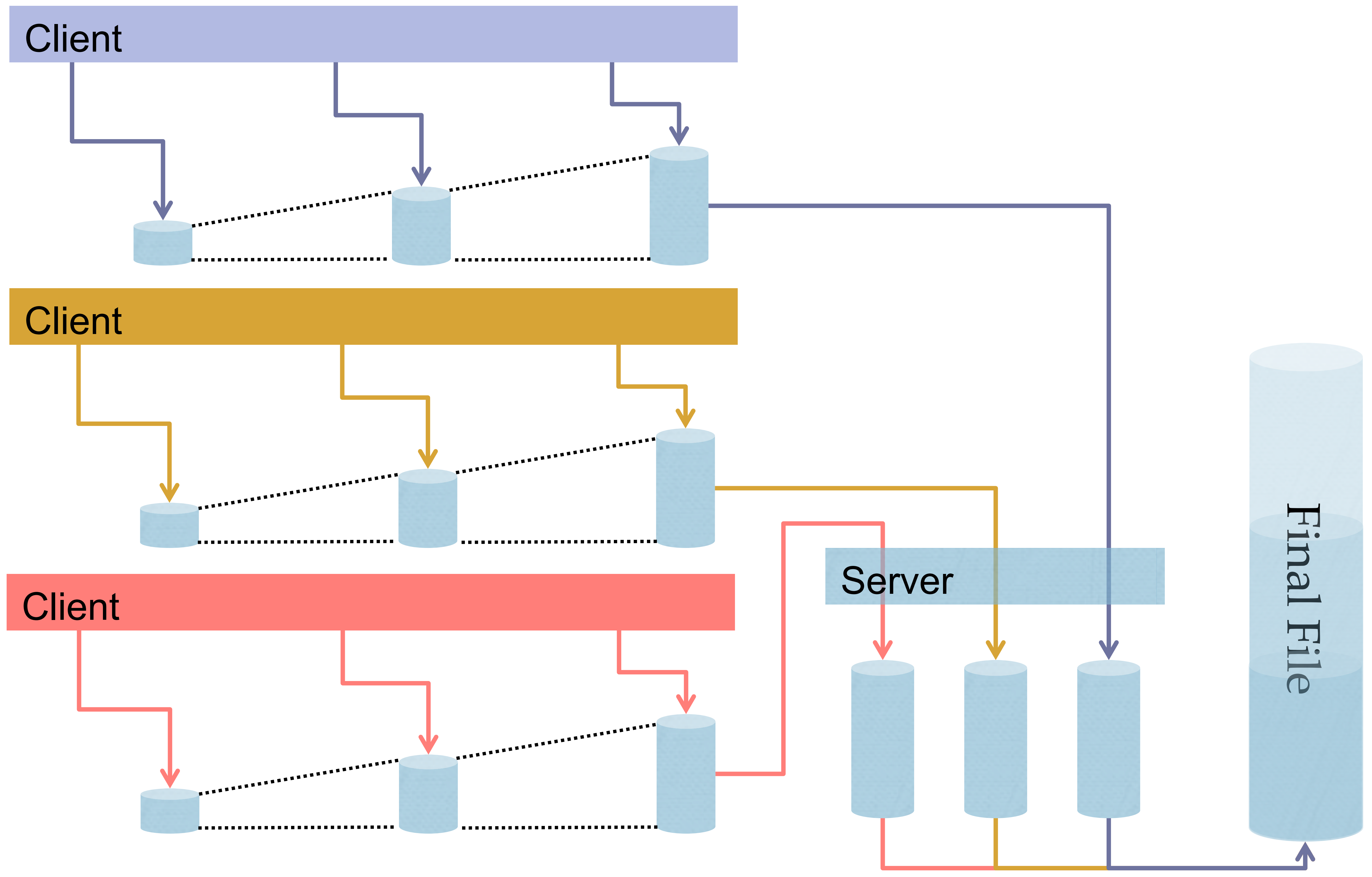
- Current effort
  - 20ish (mostly small) issues addressed

- Additional effort
  - at least 40ish (many large) issues addressed

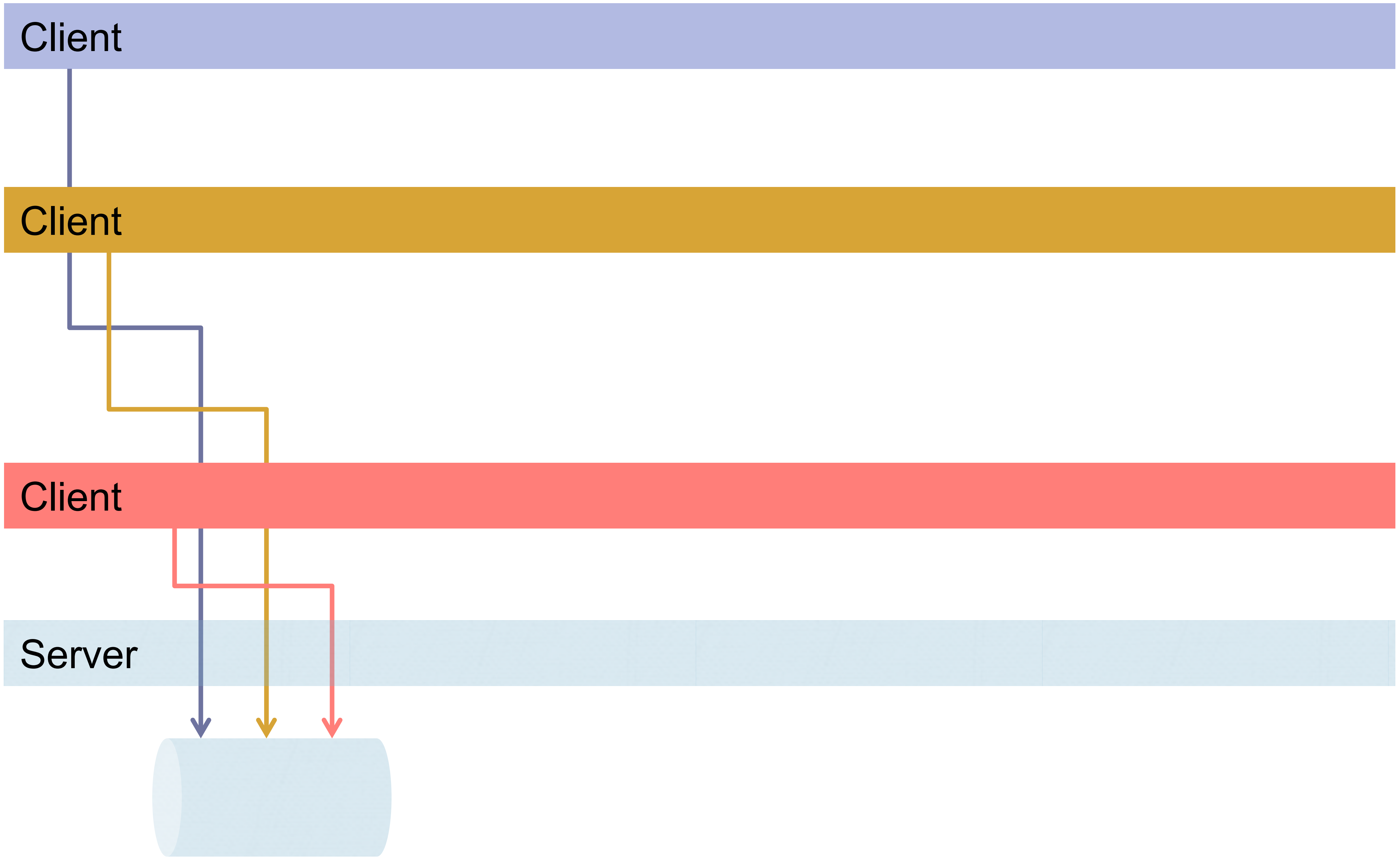
Multi Processing Bottleneck

- Number of cores and nodes increasing dramatically
- Managing very large number of files is both hard and somewhat wasteful.
- Usual solution is to merge the files.
- In addition, the number of disks is not increasing as fast
  - Hidden serialization, for example when using whole node allocation and fork on write.

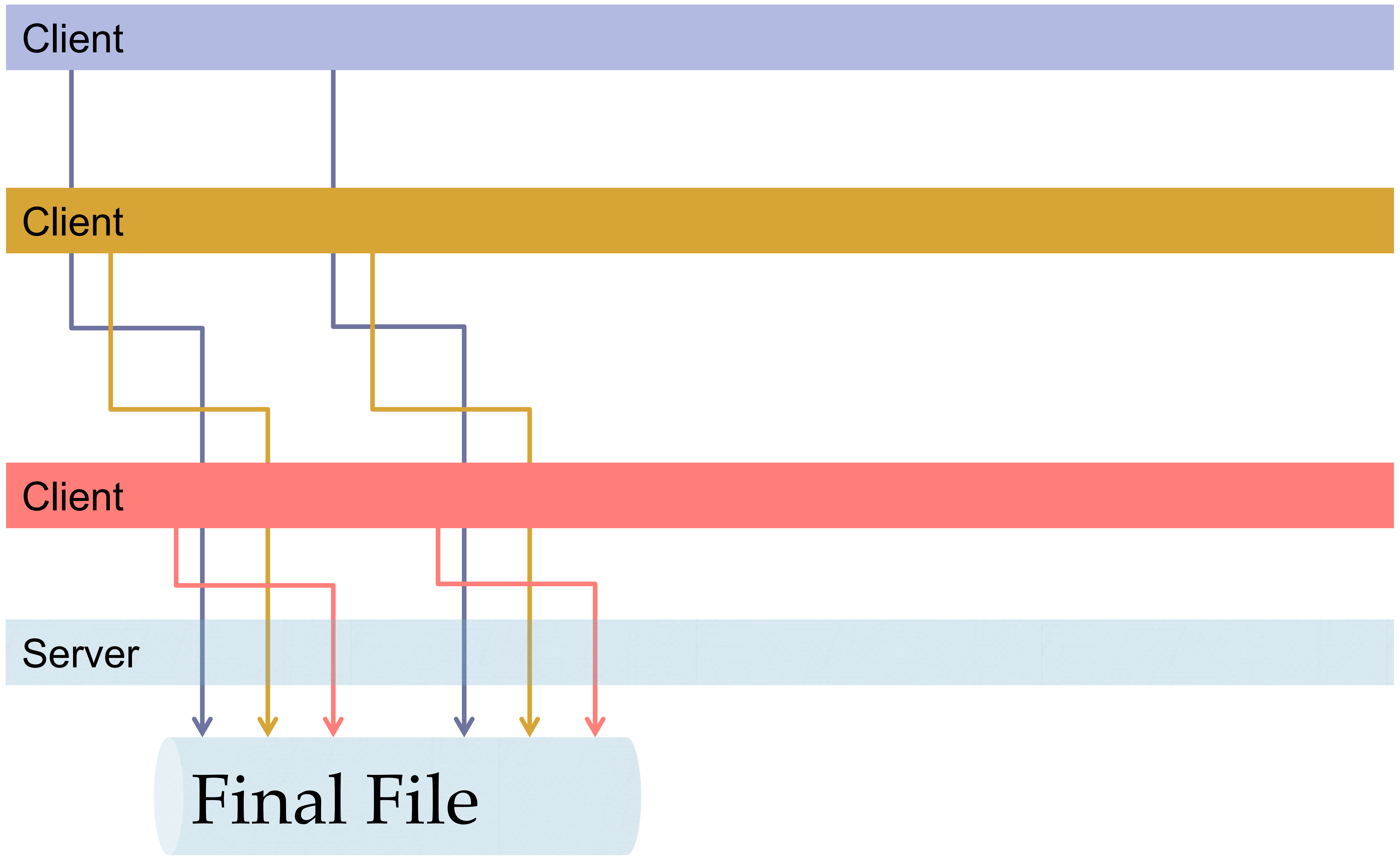
Typical Arrangement



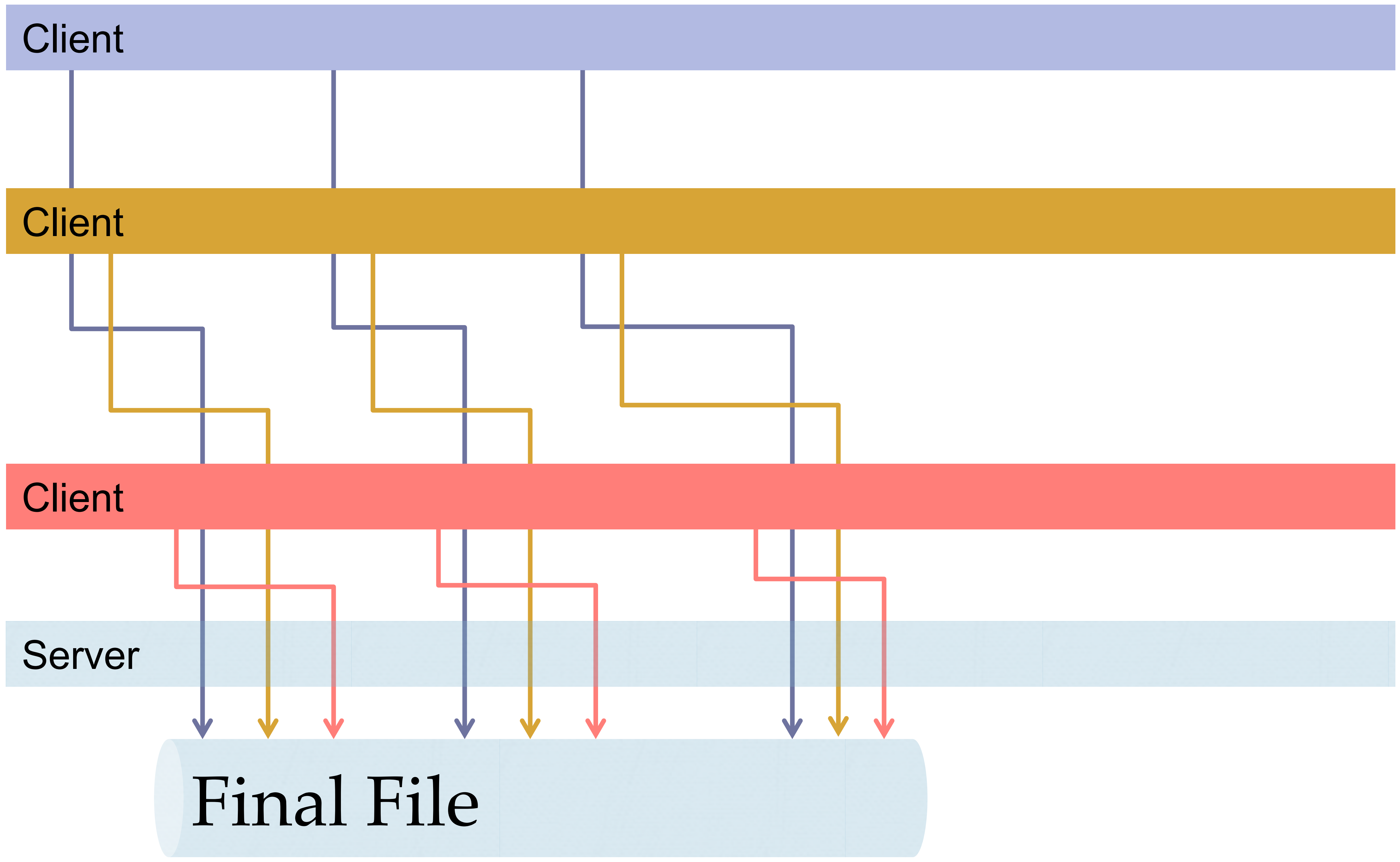
With Parallel Merging



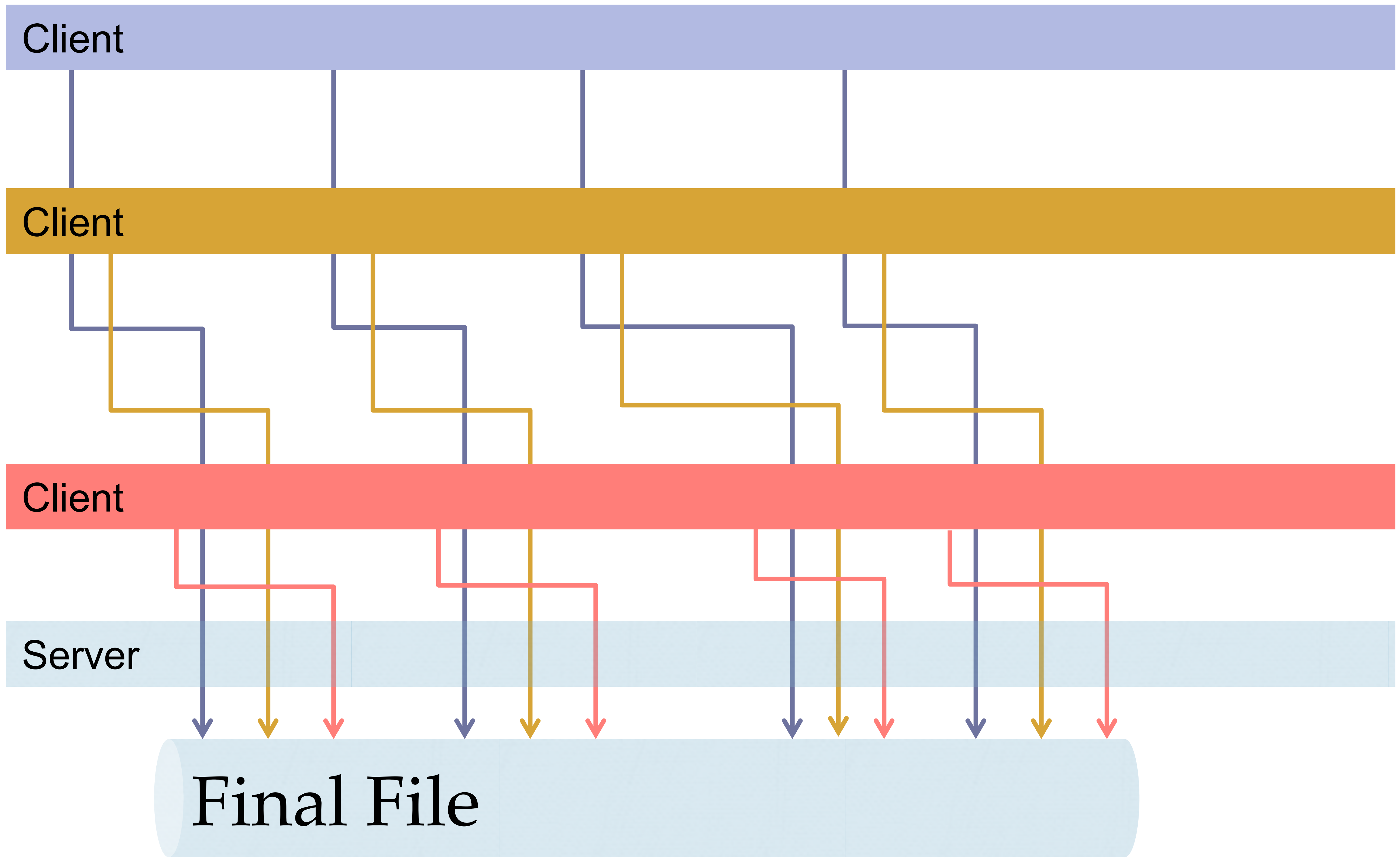
With Parallel Merging



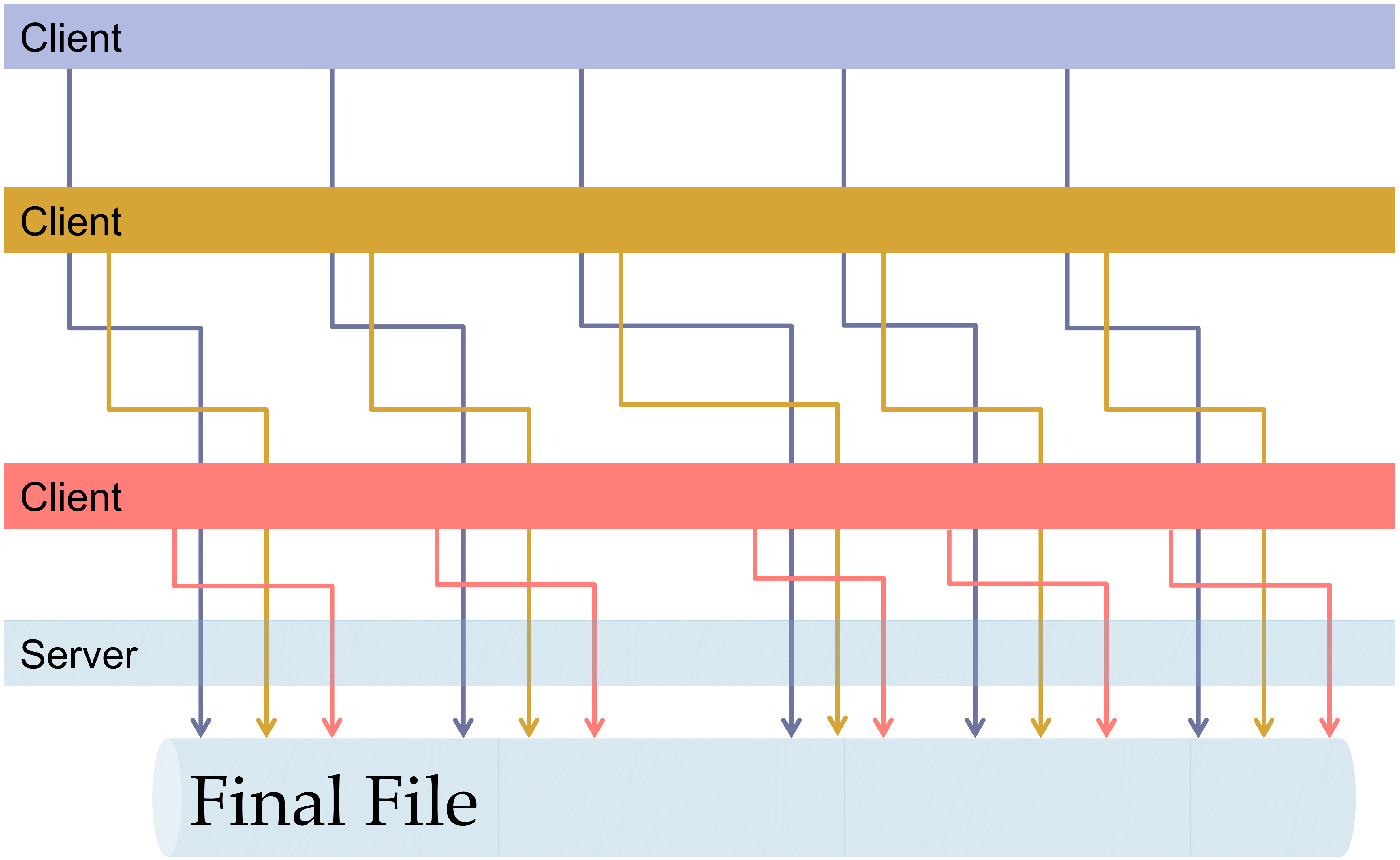
With Parallel Merging



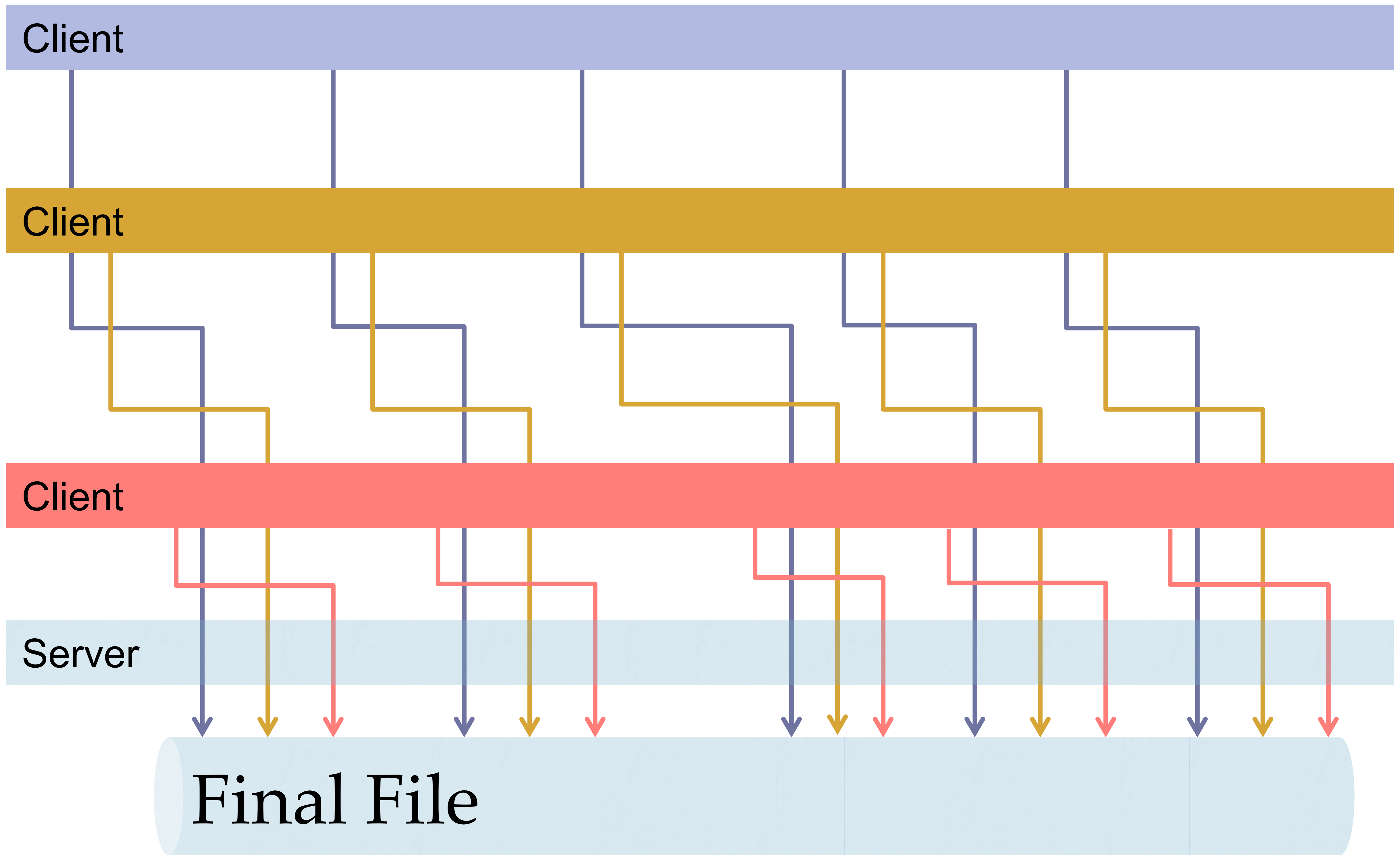
With Parallel Merging



With Parallel Merging



With Parallel Merging



TTreeReader

- New experimental interfaces to simplify and consolidate simple use cases.

```
void tread_obj() {
   // Reading object branches:
   TFile* f = TFile::Open("tr.root");
   TTreeReader tr("T");
   TTreeReaderValuePtr< MyParticle > p(tr, "p");
   TTreeReaderArray<double> e(tr, "v.fPos.fY");
   while (tr.GetNextEntry()) {
     printf("Particle momentum: %g\n", p->GetP());
     if (!e.IsEmpty())
       printf("lead muon energy: %g\n", e.At(0));
   }
   delete f;
}
```

- Automatically turns on all relevant optimizations
  – ***TTreeCache***, Partial reading. Etc.