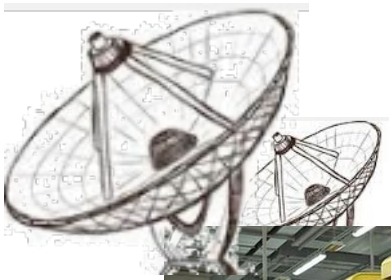
CorrelX: A Cloud-Based VLBI Correlator

V. Pankratius, A. J. Vazquez, P. Elosegui

Massachusetts Institute of Technology
Haystack Observatory

Slides based on 5th International VLBI Technology Workshop presentation
12-14 October 2016, MIT Haystack

Trends: Software-Based Instruments & Cloud Computing



Cloud Computing

- Signal Processing
- Algorithms
- Imaging
- Search & Classification
- Simulations
- Data analysis, mining, retrieval

Scalability is Key

- Examples: Current & Future Radio Arrays

  - 10s of Tbit/sec during cross-correlation
  - 1 terabyte/day typical output data rate

Raw Data		Cost	Sensors
~72 Tbit/s	ASKAP (under construction)	$160M	96-pixel phased array feeds on 36 12m dishes
~4 Tbit/s (10 PB science-ready data / year)	MeerKAT (under construction)	$100M	64 13.5m antennas, wideband single-feed pixels
~10 Tbit/s	SKA Phase 1 (construction starting 2017)	$900M	low-freq. component: 262144=2^{18} dipoles
~2.5 Pbit/s	SKA Phase 2 (construction ?)	?	3-4 million antennas

CorrelX

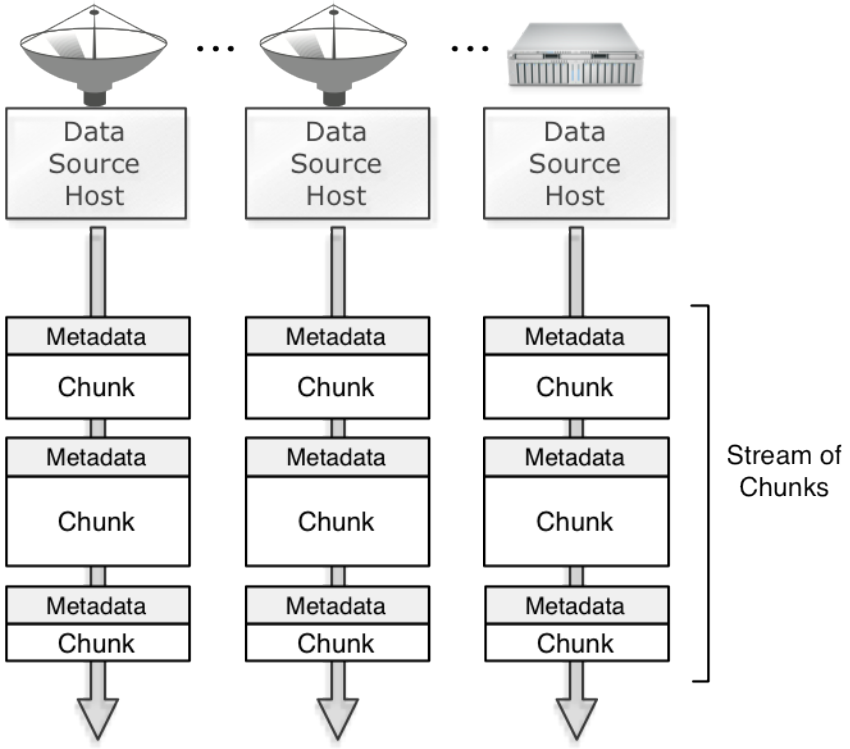
New architecture

- Breaks with current paradigms that aim to mimic hardware correlators in software
- Based on chunk streaming model
- Enables offline, out-of-order correlation of chunks

Key Features

- Cloud scalability
- Elastic correlator deployments for variable workloads, burst capability
- Reliability on large number of nodes through built-in online testing mechanisms (injected control chunks)
- Plugin-based. Enables quick prototyping and extensions for research and teaching
- Separate concerns: core correlation code vs. parallel computing code
- Leverages open-source: Map-Reduce, parallel file systems
- Enables machine-learning-based performance optimization

CorrelX: Chunk Streaming Model



- Chunks introduce locality → out-of-order parallel processing
- Enables tunable chunks (sizes, sampling, numerical precision, etc.)
- Insert control chunks with known results for validity checks on heterogeneous or unknown hardware

CorrelX

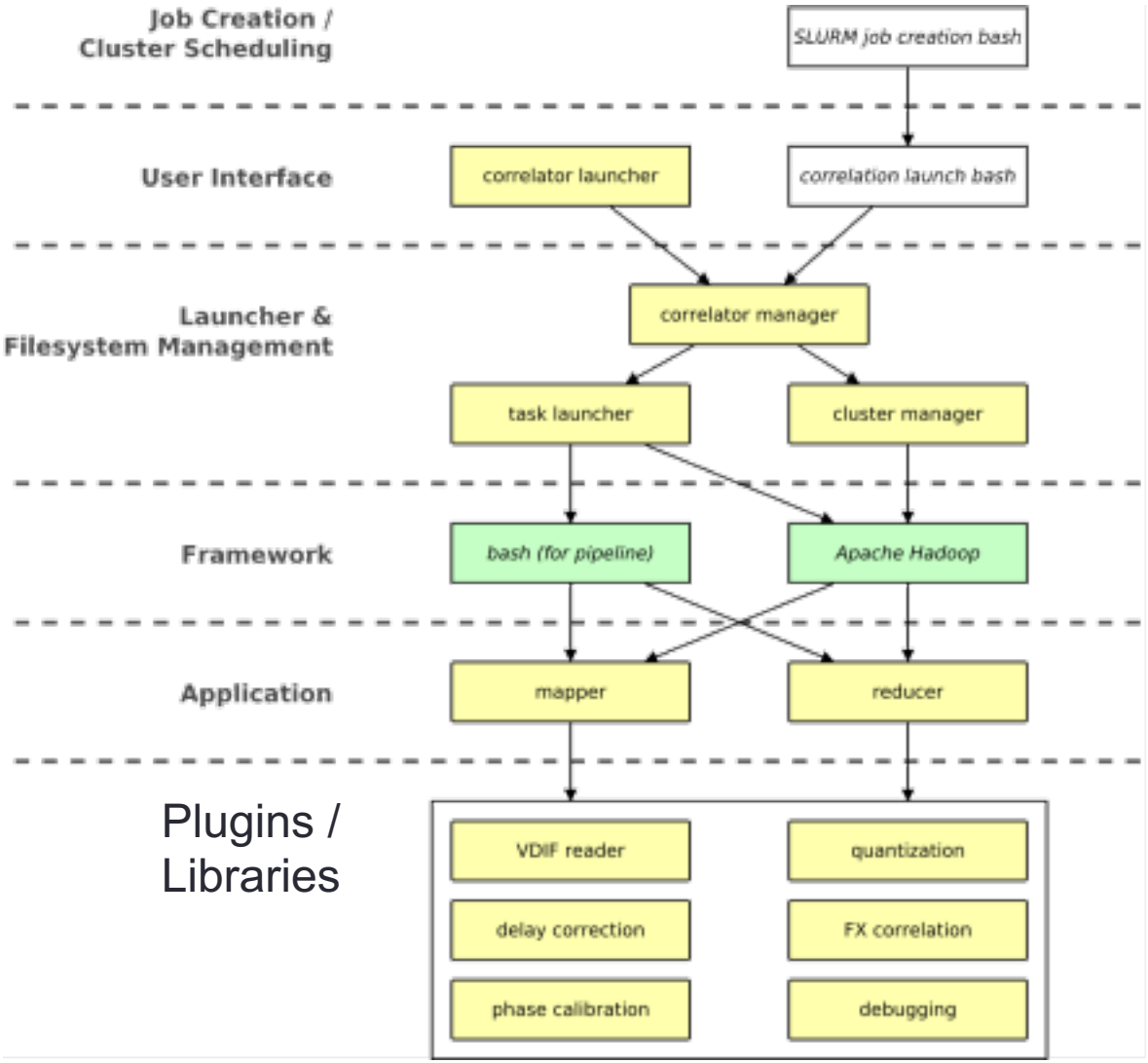
- Proof of concept prototype, incremental feature additions
- Core: Map-Reduce paradigm
  - Popularized by Google
  - Programming model: two types of workers:
    - Map(input_block) → set of [key,value] pairs
    - Reduce(set of [key,value] pairs) → set of [results]
  - Parallelism hidden by framework:
    - Framework launches one map per input block, one reducer per partition (i.e., all pairs identified by same key).
  - Apache Hadoop: mapper and reducer can be written in different languages (e.g., Python, C++, ...)
  - Parallel file system: Lustre
    - Max. volume size: "over 16 EB (theoretical)" (2.8.59, Oct 2016)
    - Used in over 50% of Top 100 supercomputers
    - "Presents all clients with a unified namespace for all of the files and data in the filesystem"

CorrelX

- CorrelX Correlation Configuration
  - Deployment and performance configuration: .xml files
  - Partitioner: define sub-keys for partitioning (what goes into each reducer) and sorting (to reconstruct "order" after "out-of-order" parallel computing)
- FX correlation
  - Parallelization in frequency bands and accumulation periods
  - Parallelization in baselines configurable: single-baseline-per-task, linear-scaling-stations and all-baselines-per-task (default is all-baselines-per-task)
- Delay correction using polynomials from CALC (vex2difx+calcif2).
- Phase calibration tone extraction
- Zoom bands (currently during post-processing)
- Interfaces:
  - Input format: VDIF (configuration read from frame header), complex or real, 1 or 2 bits per sample, multiple threads xor multiple channels per frame
  - Output format: text file with visibilities and phase calibration results

CorrelX Implementation Details

- Correlator manager
  - Deploy cluster
  - Process configuration files
  - Launch correlation
- Cluster manager
  - Methods for starting/stopping cluster
  - Methods for filesystem initialization
- Task launcher
  - Methods for correlation job launch
- Map
  - VDIF read and corner-turning
  - Initial integer-sample alignment
- Reduce
  - Fractional sample shift
  - Fringe rotation
  - Fourier transform
  - Fractional Sample Correction
  - Multiply and accumulate



CorrelX Implementation Details

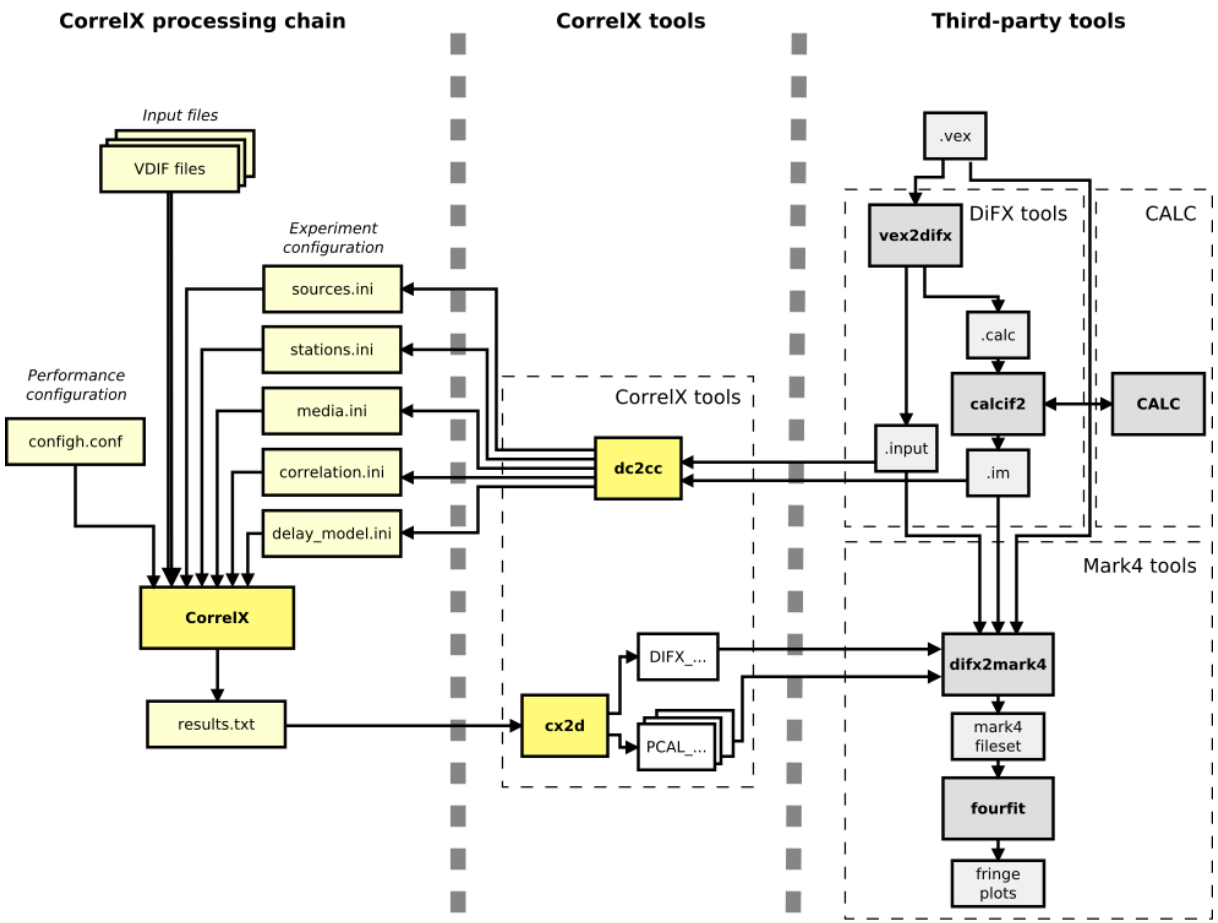
Project	Language	Lines of code
CorrelX	Python	9k 2k comments
Libs.	Python	2M
Hadoop	Java	2M
Lustre	C	1M

This is code maintained by open-source projects

CorrelX Layer	LOC (approx.)
Launcher and FS Management	2k
Application (map, reduce)	1.8k
Libraries (VDIF, quant., FX…)	4k
Tools (dc2cc, cx2d…)	1.2k

This is the code the VLBI community needs to maintain

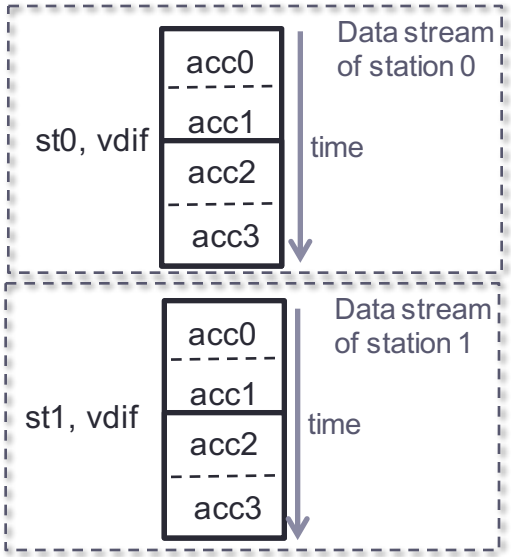
Tool Chain Integration



- Tools for integration with existing processing chains:
  - Input file converter from .input and .im
  - Output file converter to SWIN and PCAL files
- Integration with Mark4 post-processing chain
  - dc2cc: configuration
  - cx2d: output conversion
- Experiment files
  - sources.ini
  - stations.ini
  - media.ini
  - correlation.ini
  - delay_model.ini
- Correlator configuration
  - configh.conf

Correlation Example

- Correlation example:
  - Cluster:
    - 3 nodes.
    - 2 cores/node.
  - Experiment:
    - 2 stations.
    - 2 files, 2 parts/file
      => 2*2= 4 mappers
    - 2 bands.
    - 4 accumulation periods
      => 2*4=8 reducers



Mapreduce description

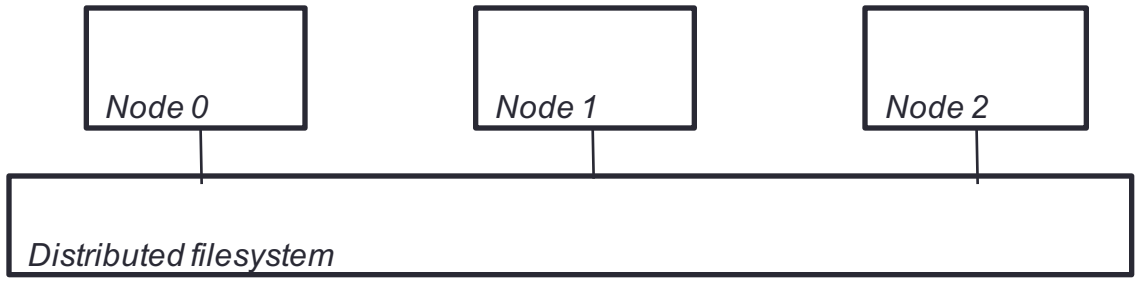
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

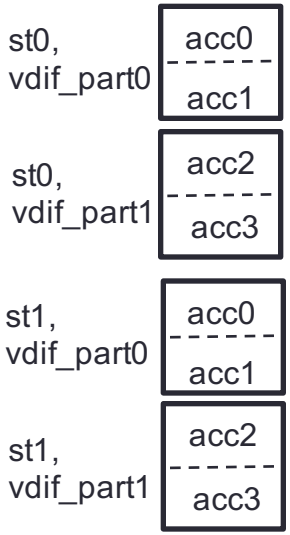
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.



Mapreduce description

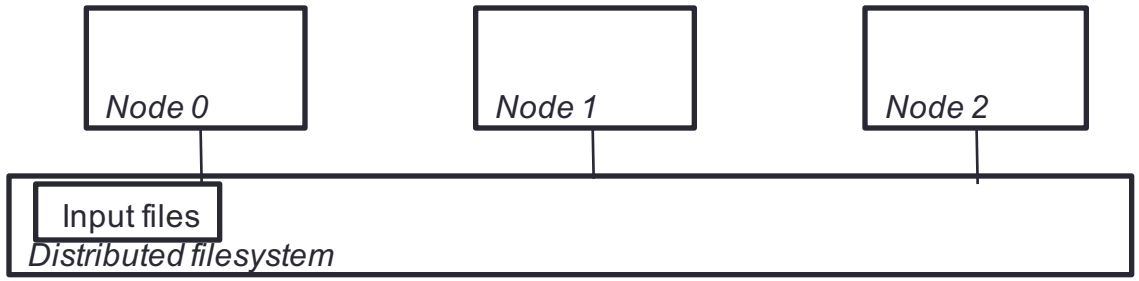
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

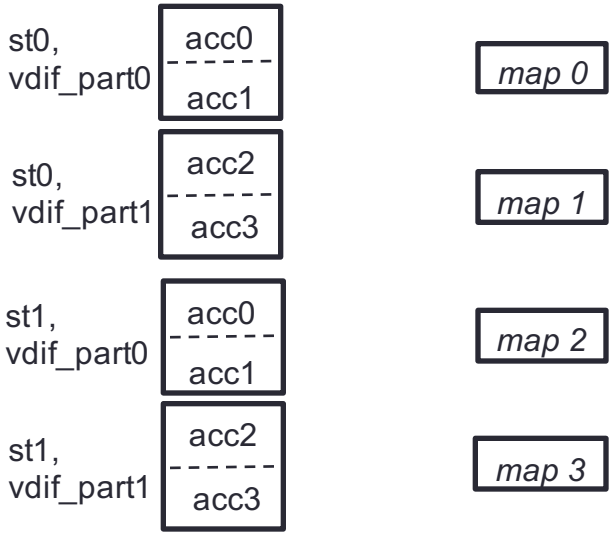
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.



Mapreduce description

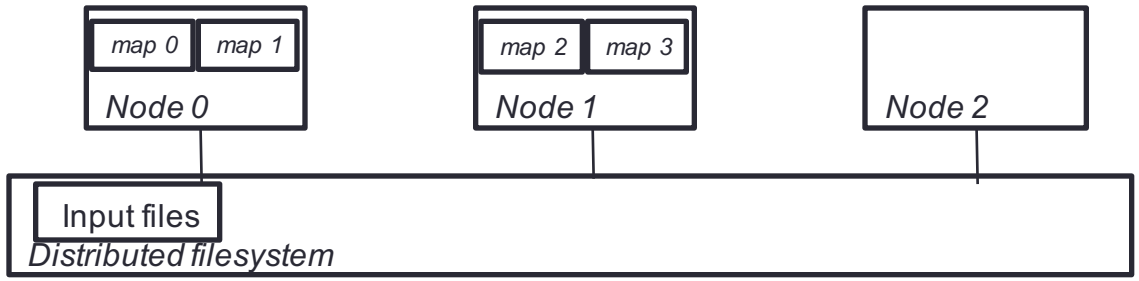
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

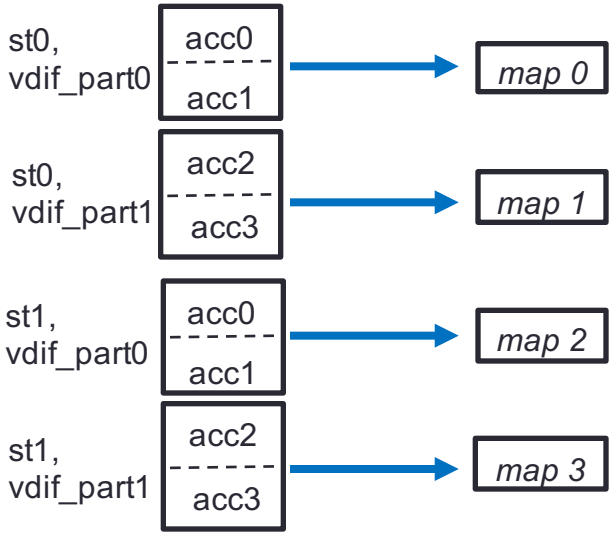
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.



Mapreduce description

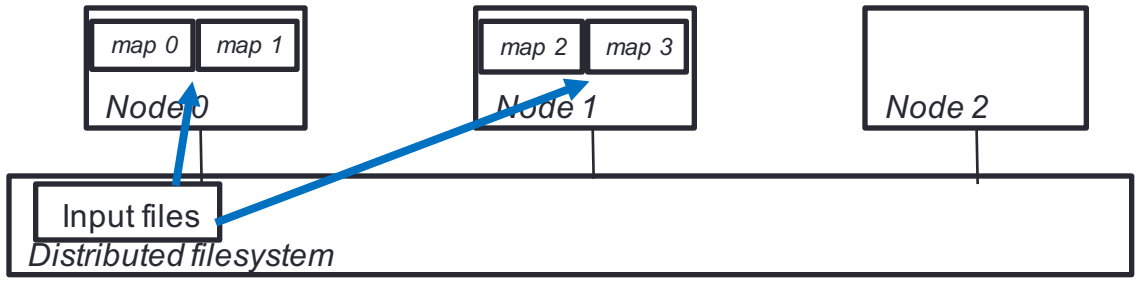
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

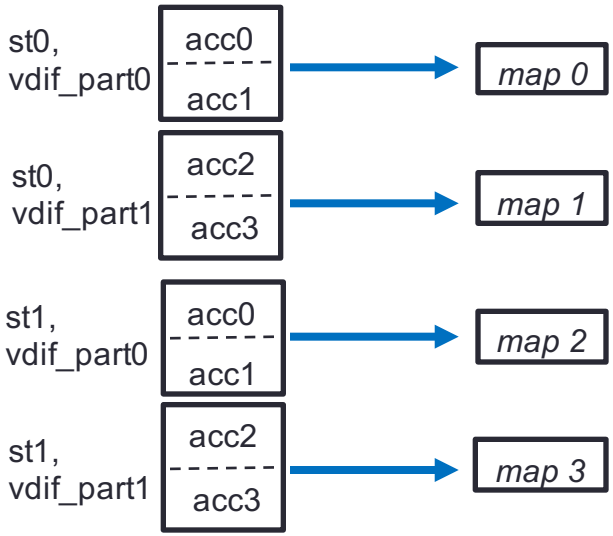
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.



Mapreduce description

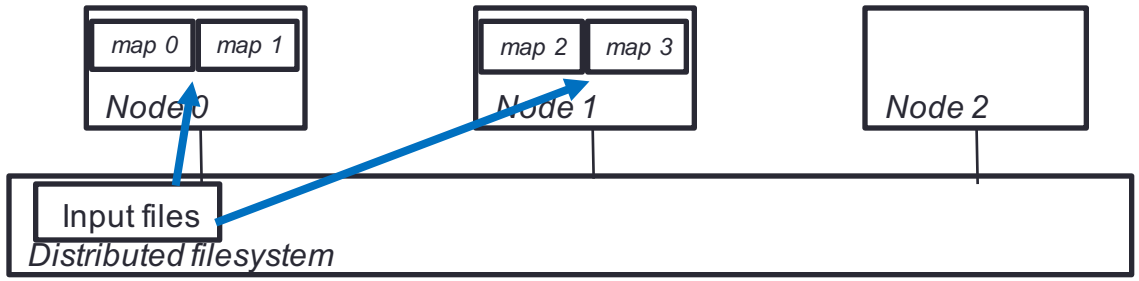
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

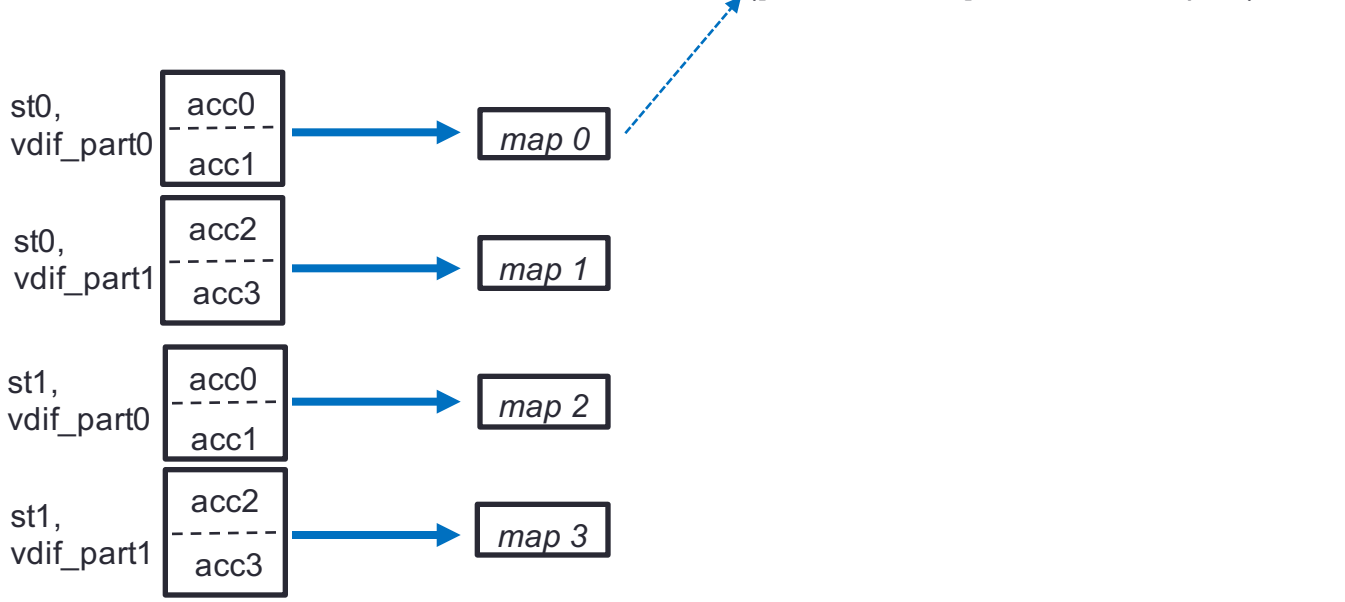
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

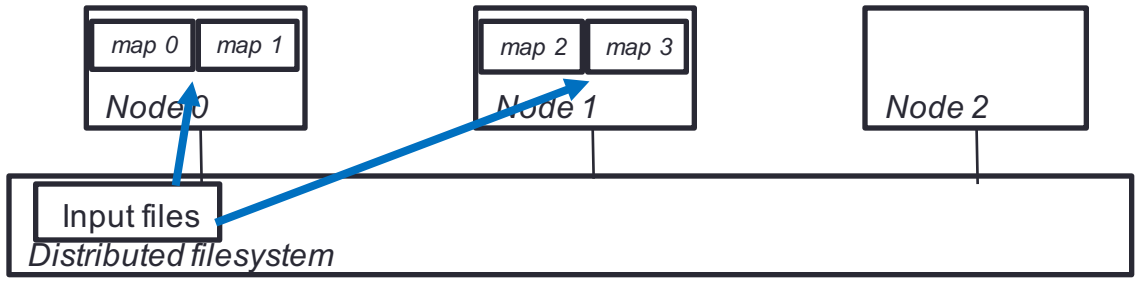


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

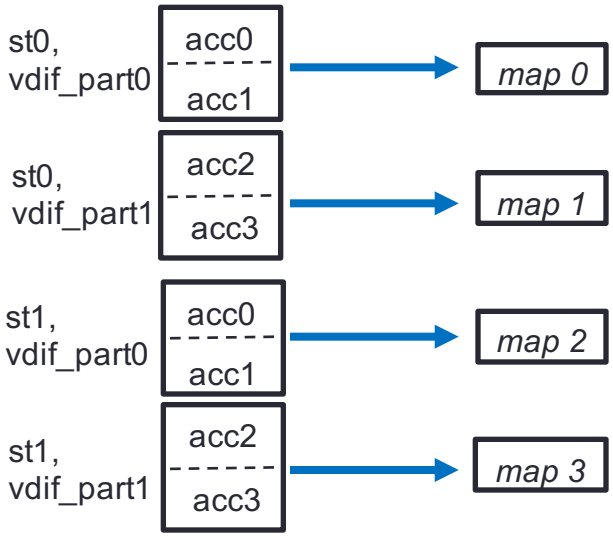
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.



Mapreduce description

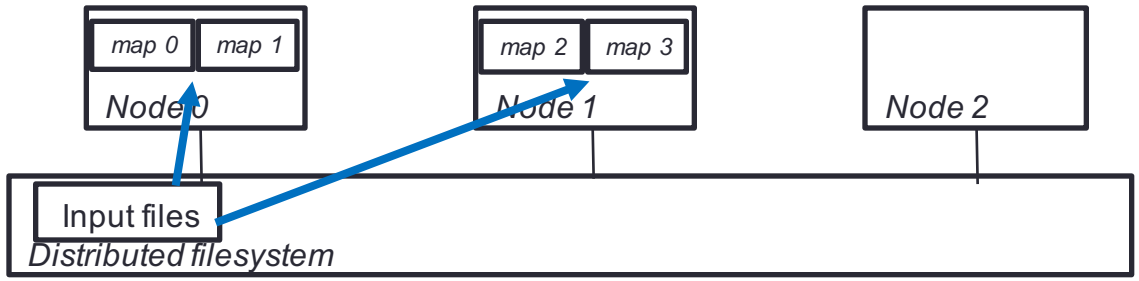
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

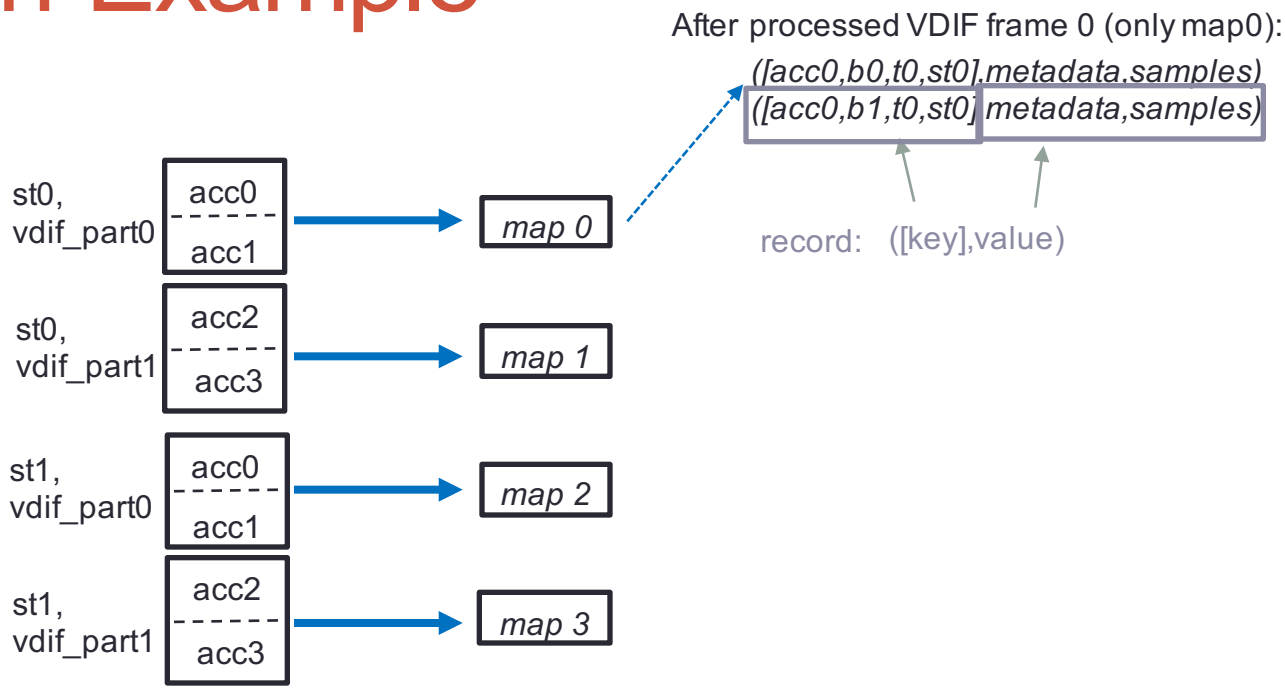
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

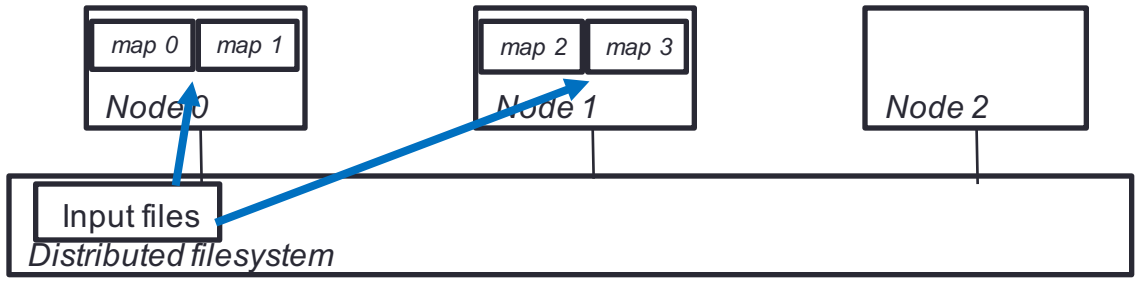


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

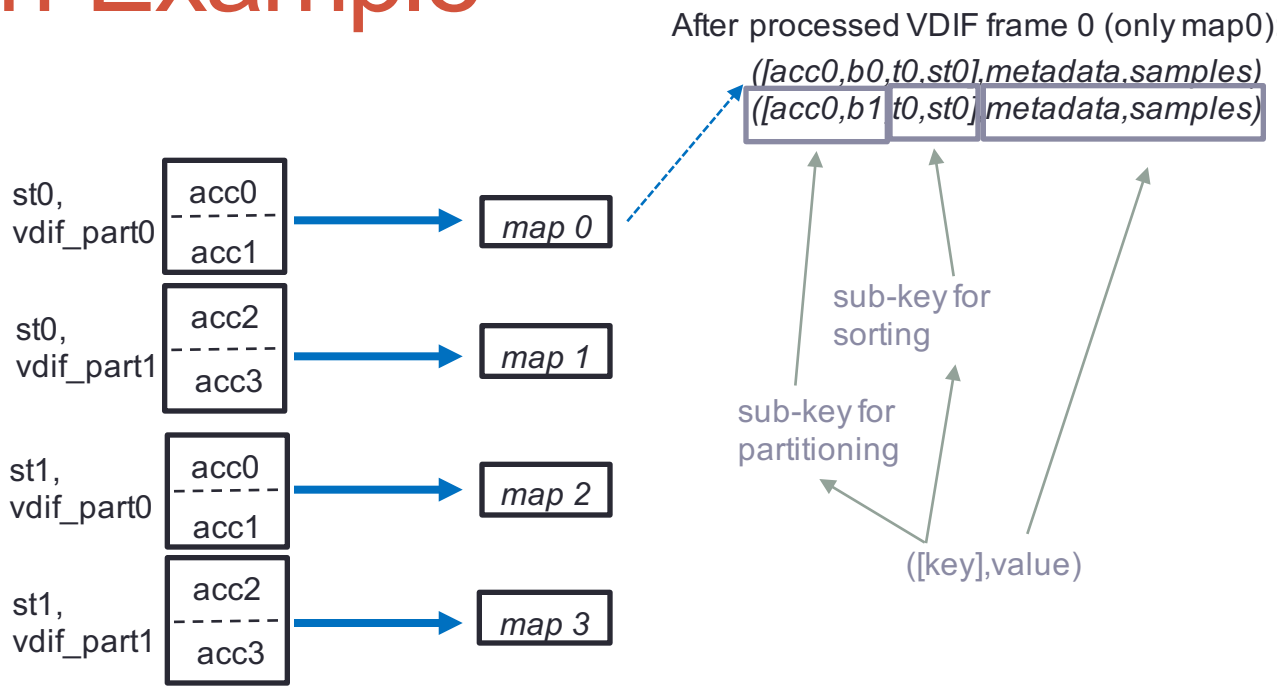
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.



Mapreduce description

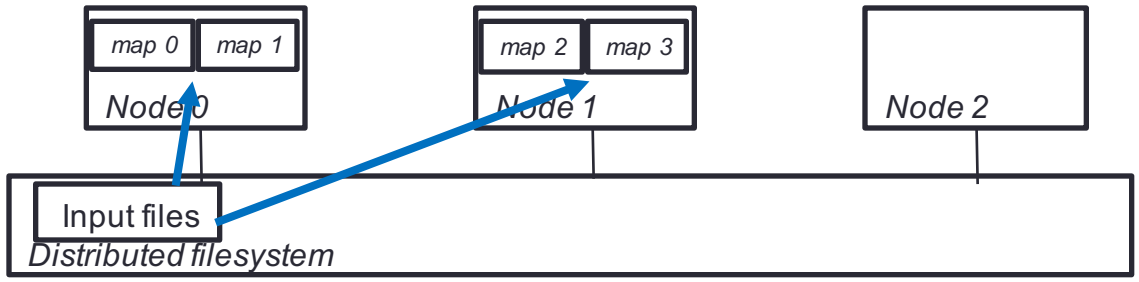
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

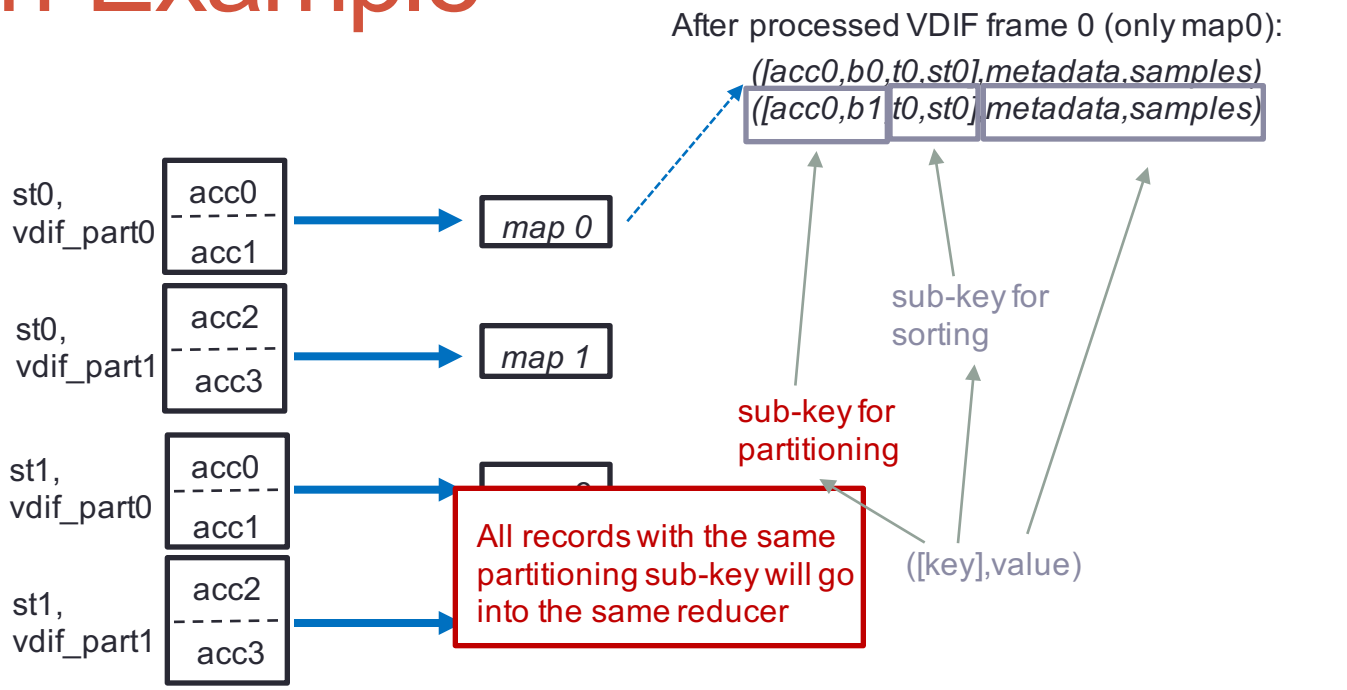
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

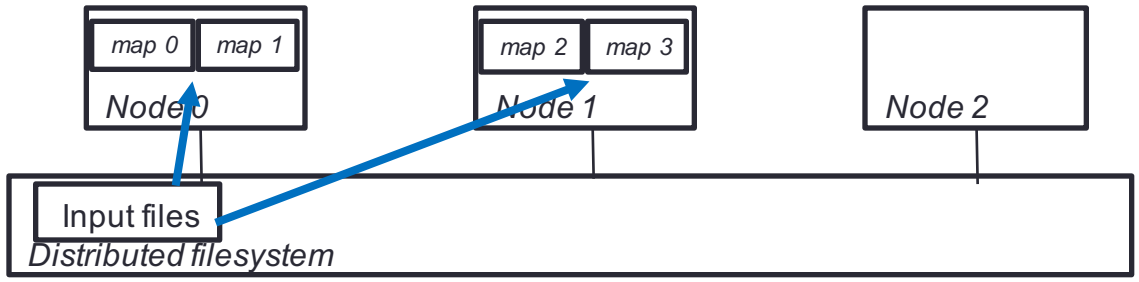


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

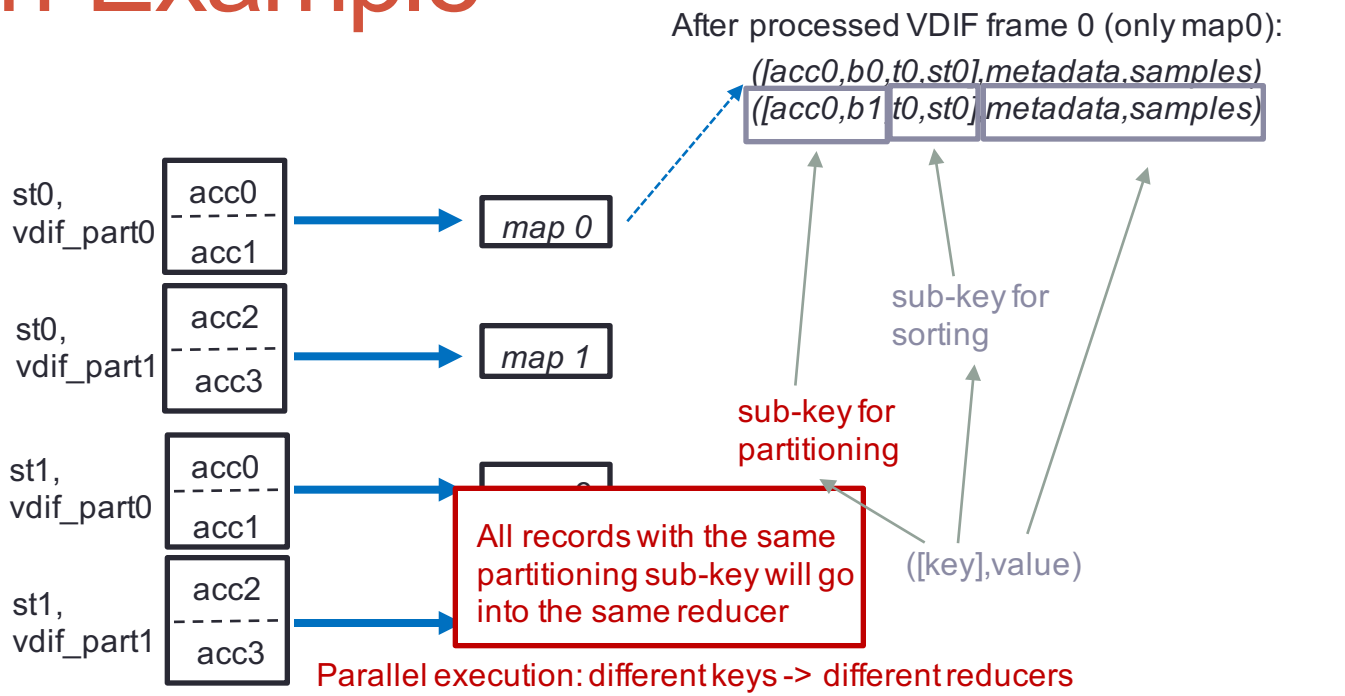
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

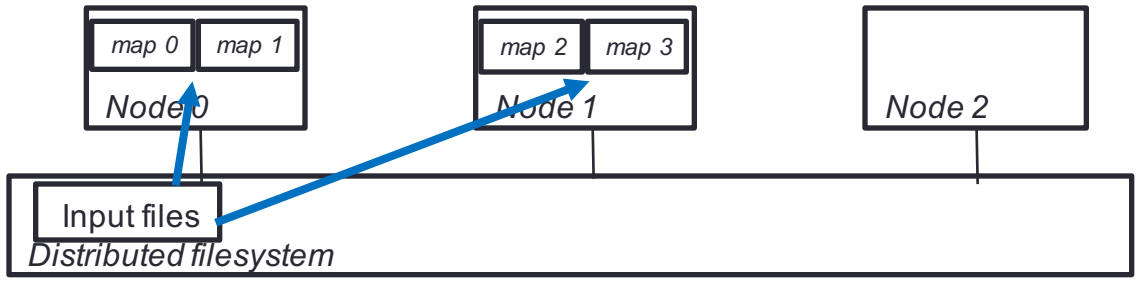


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

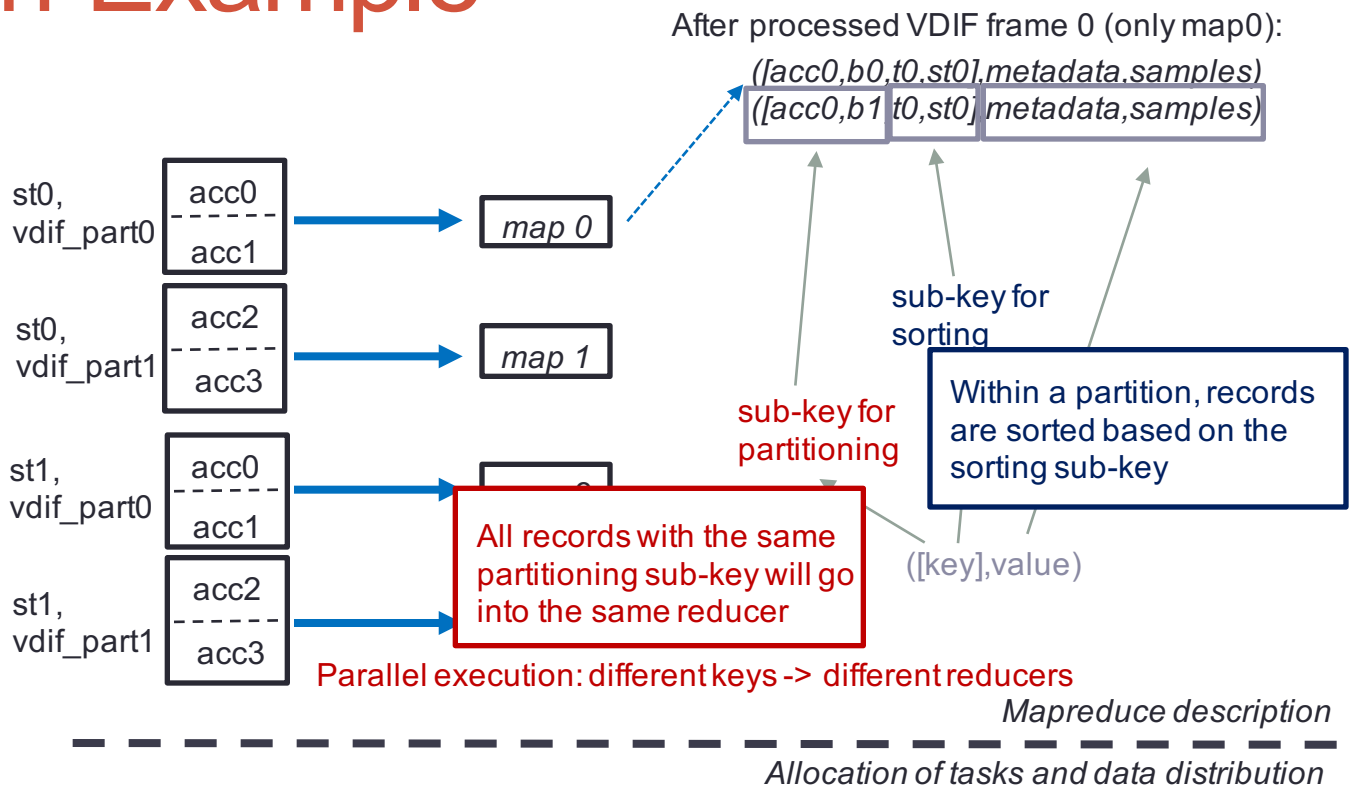
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

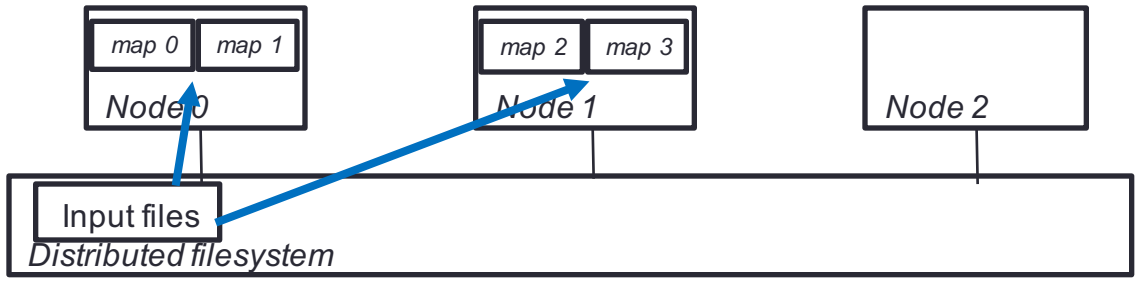


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

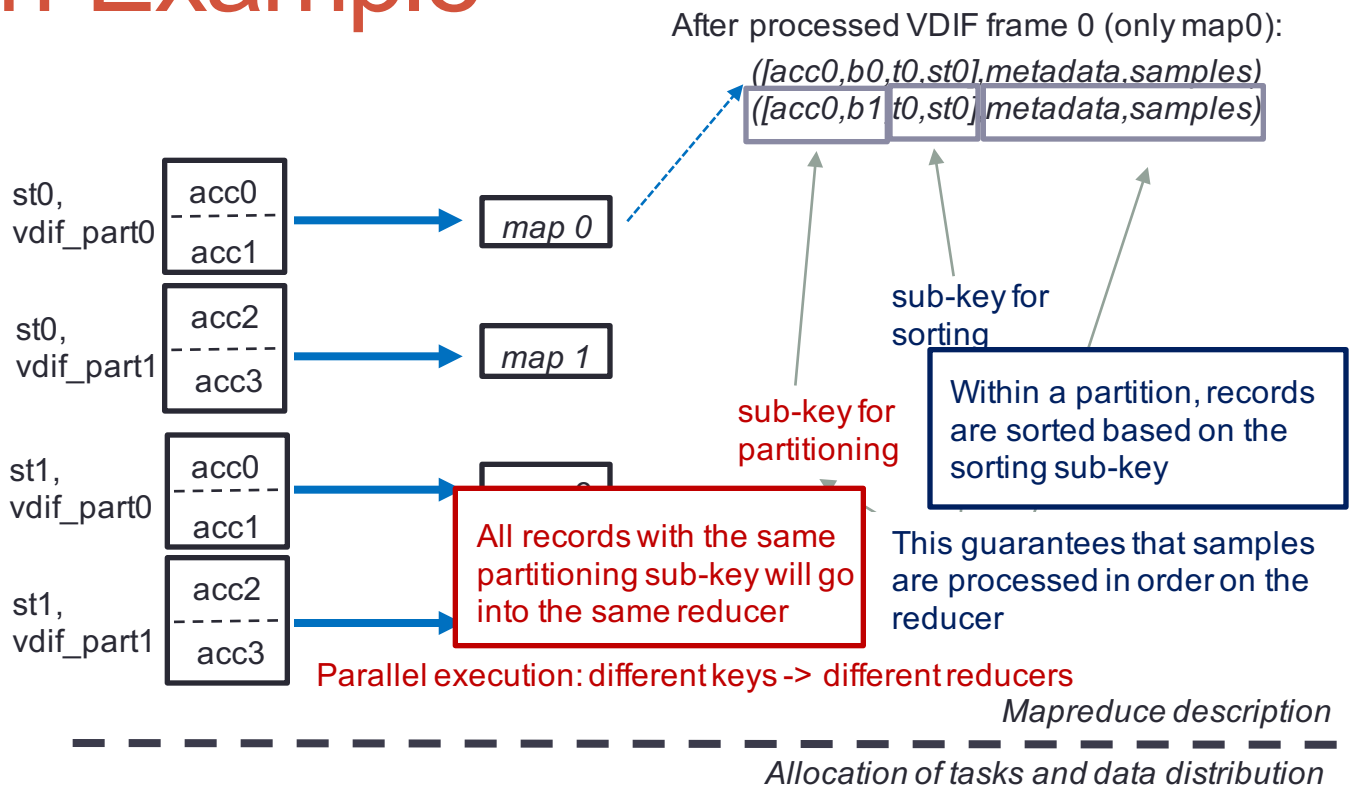
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

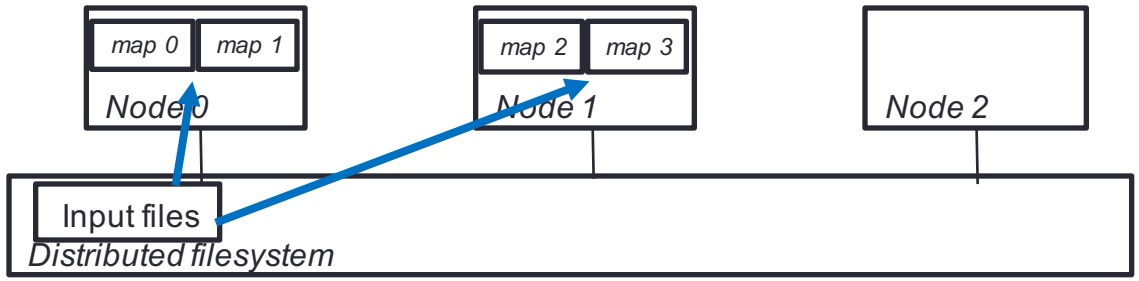


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

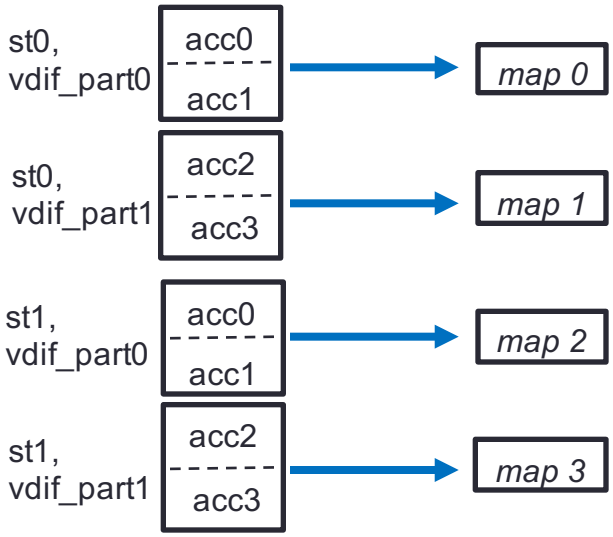
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.



After processed VDIF frame 0 (only map0):

([acc0,b0,t0,st0],metadata,samples)
([acc0,b1,t0,st0],metadata,samples)

Mapreduce description

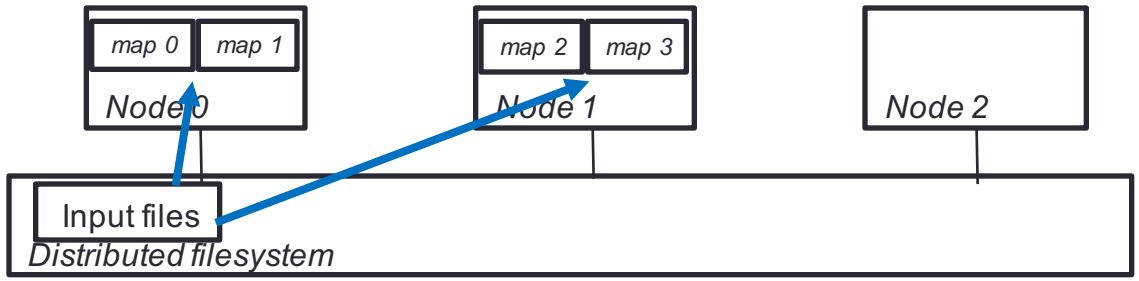
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

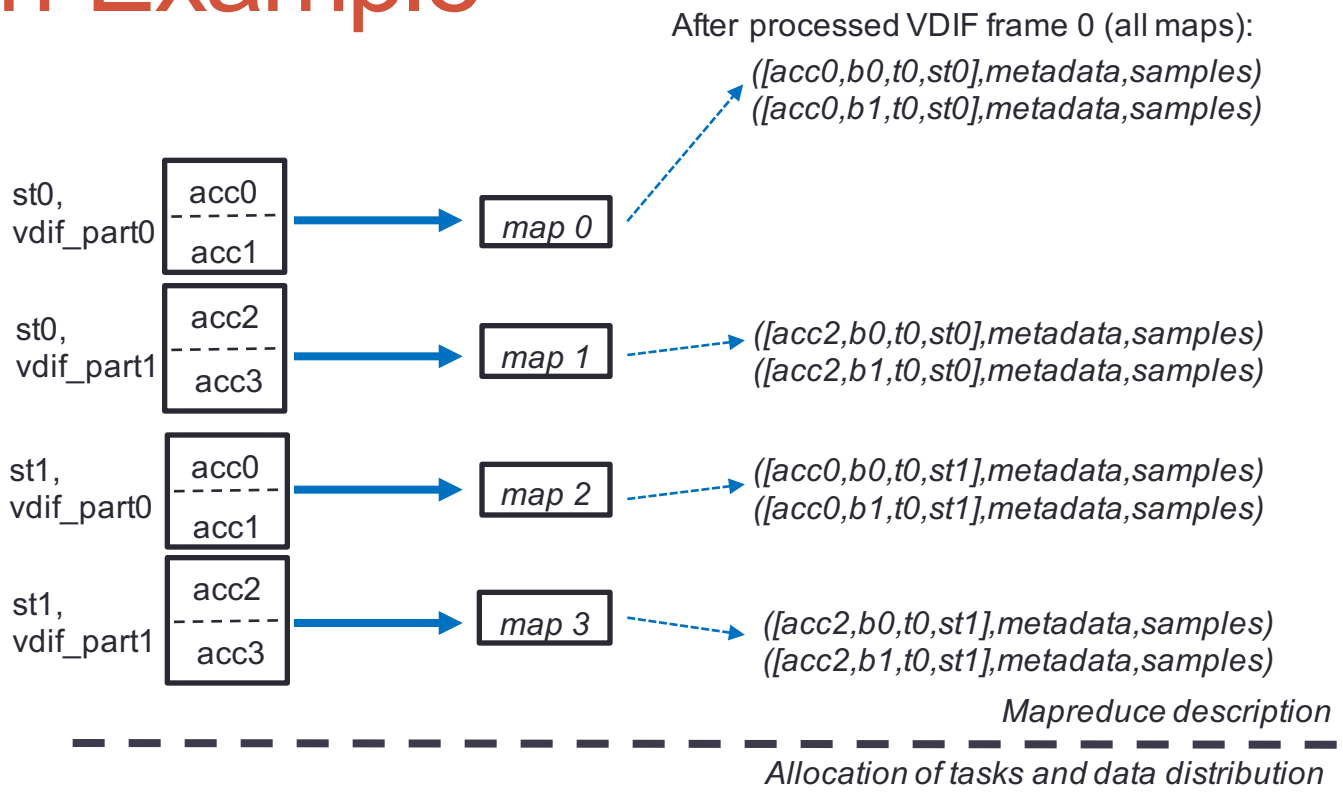
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

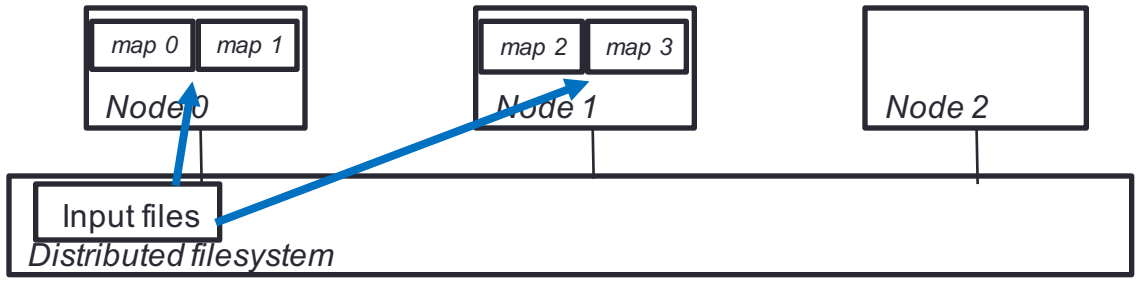


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

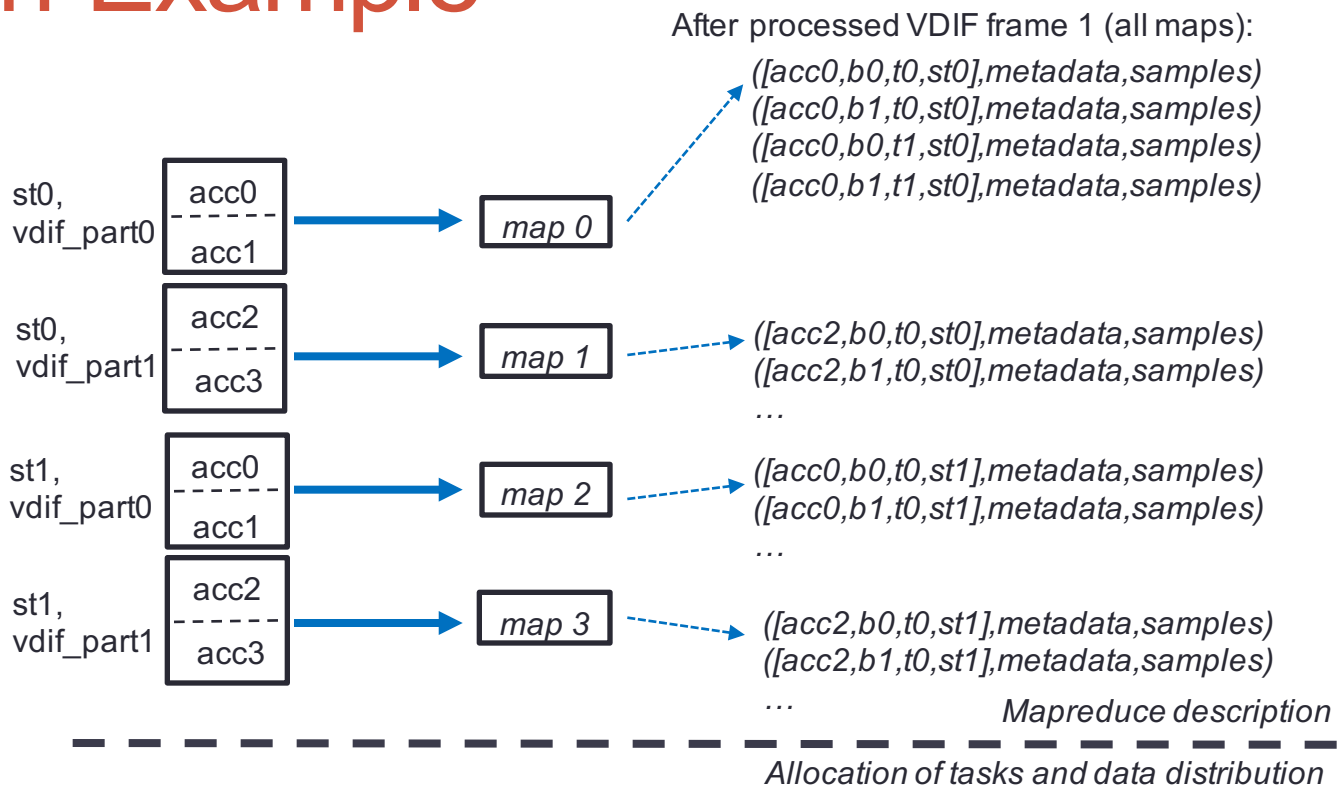
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

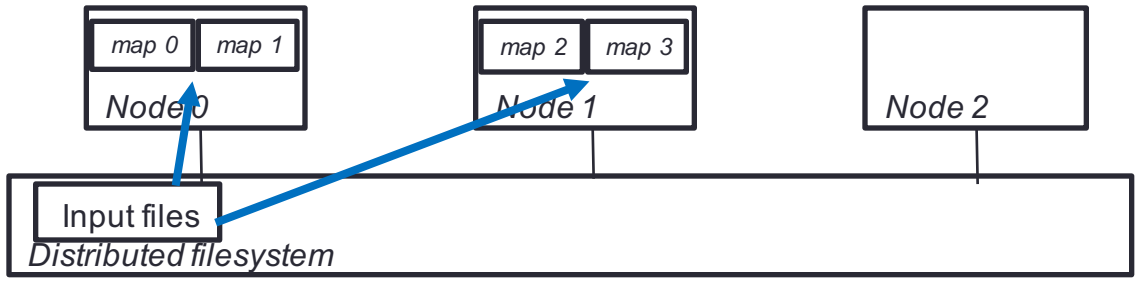


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

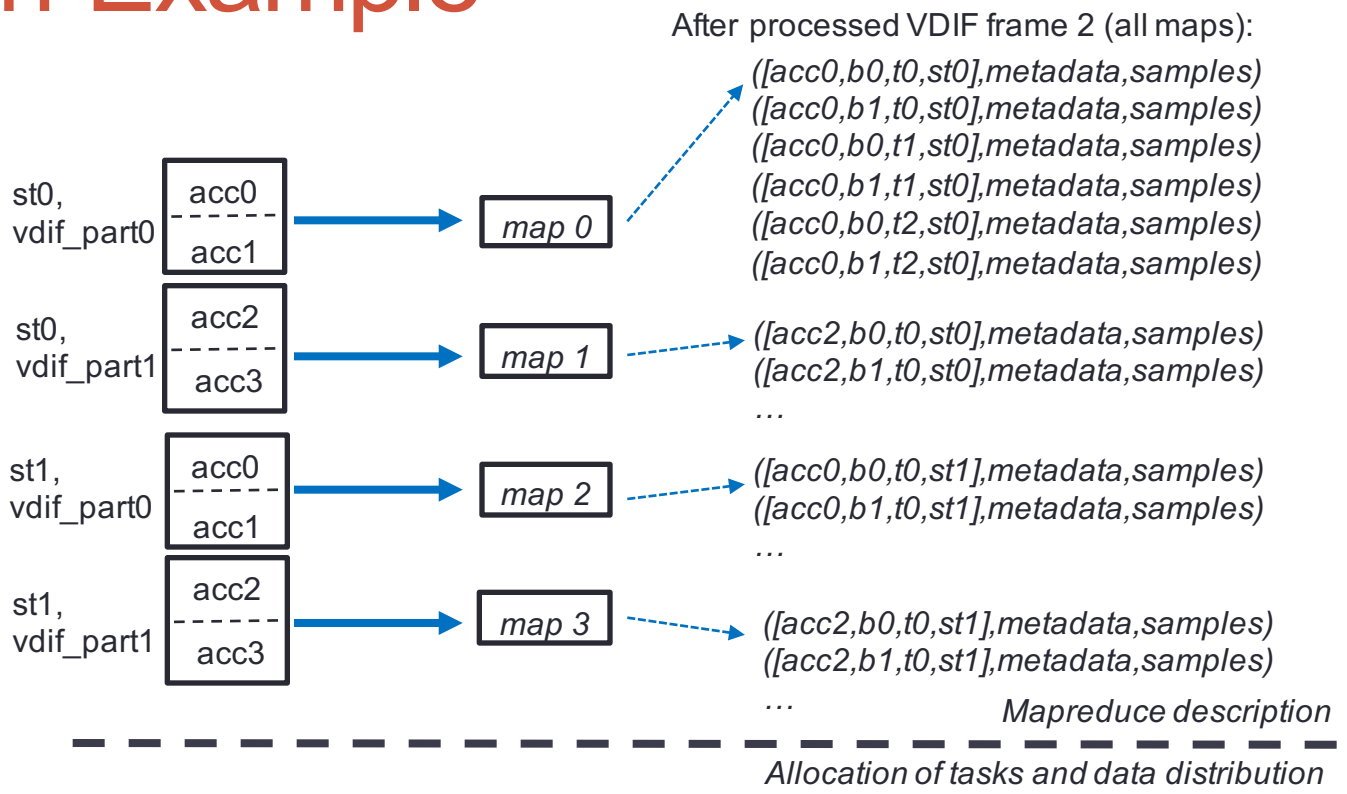
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

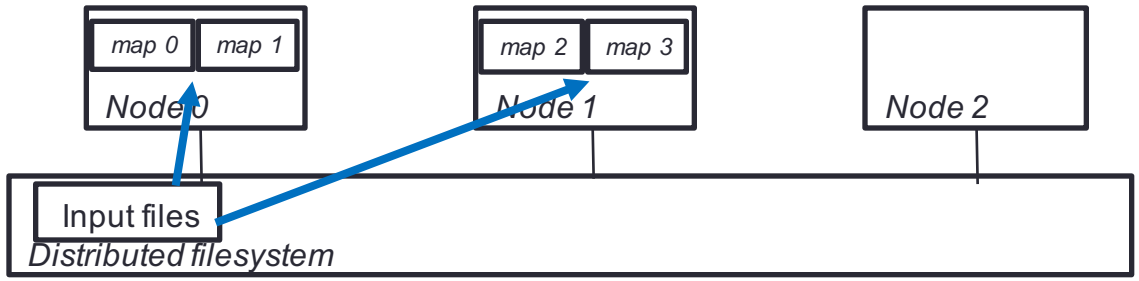


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

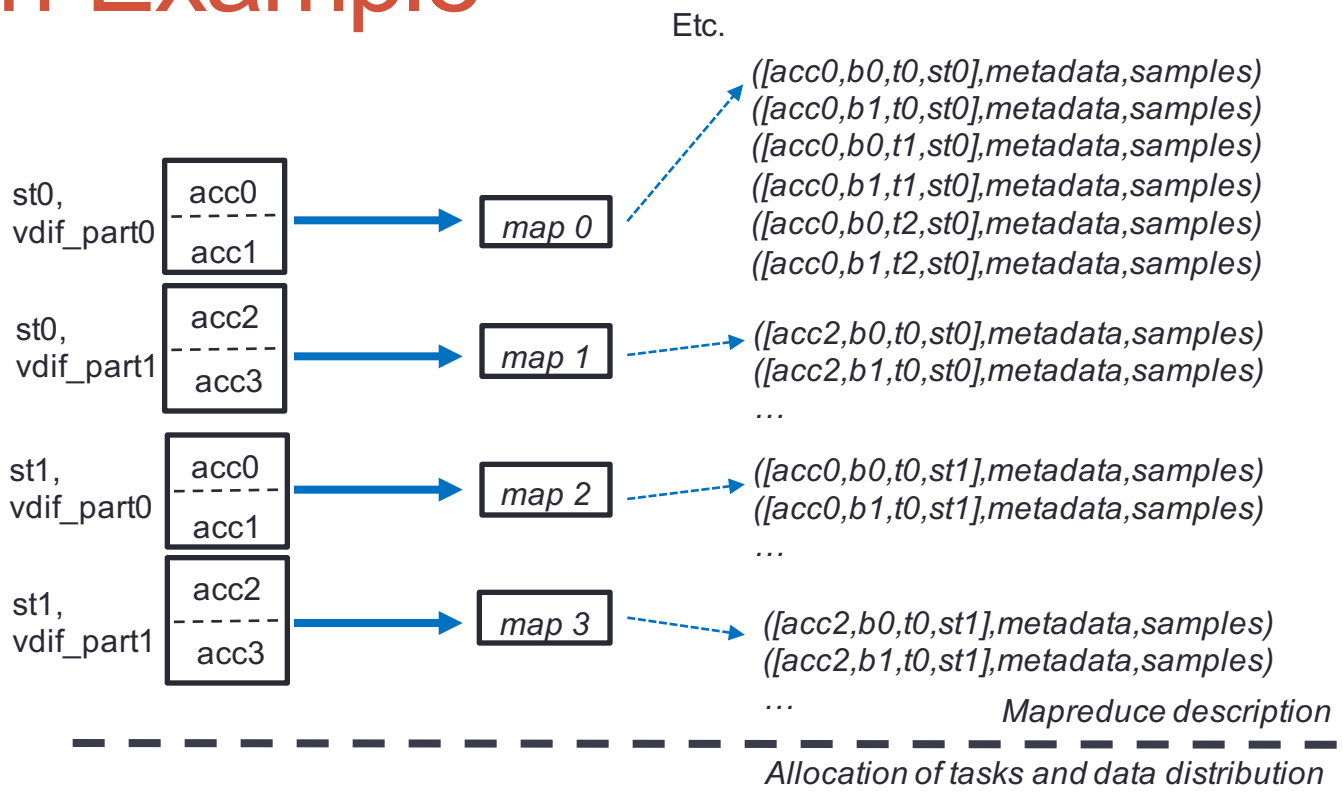
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.

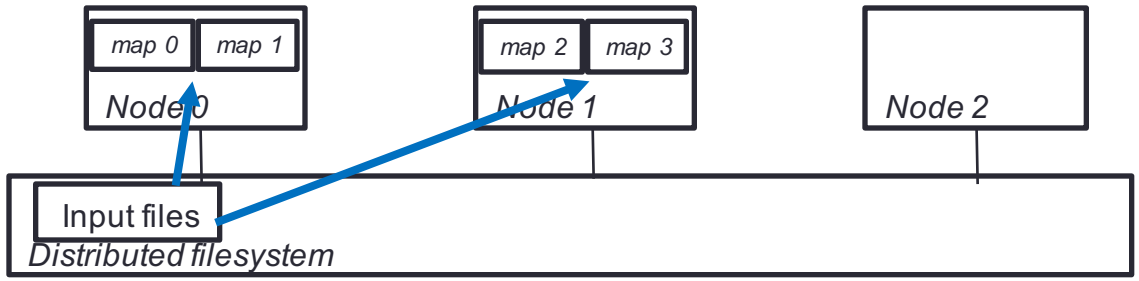


MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

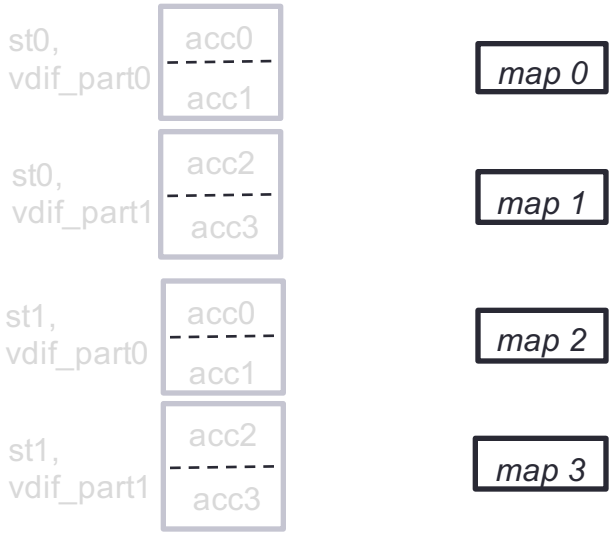
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.



Mapreduce description

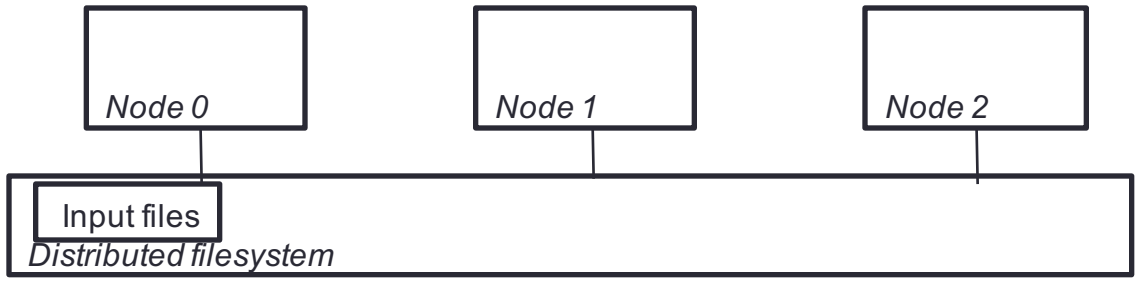
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

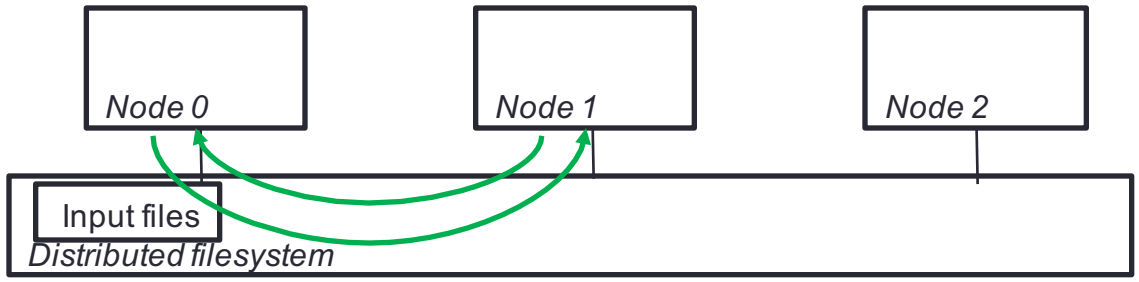
1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.

Mapreduce description

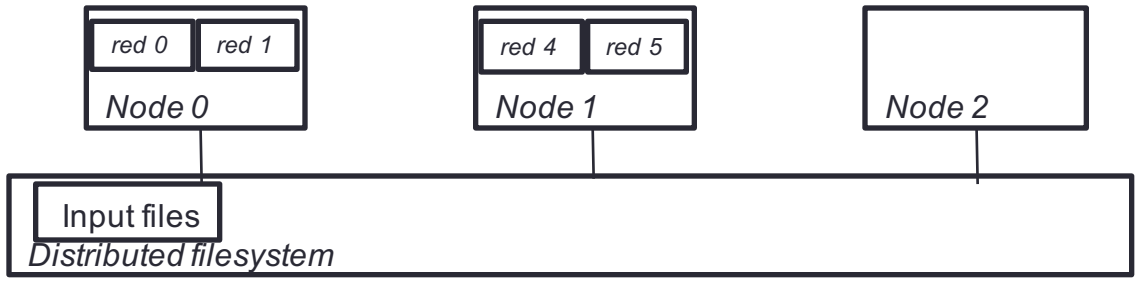
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

-Fractional sample shift.
-Fringe rotation.
-Fourier transform.
-Fractional sample correction.
-Multiply and accumulate.

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.

Mapreduce description

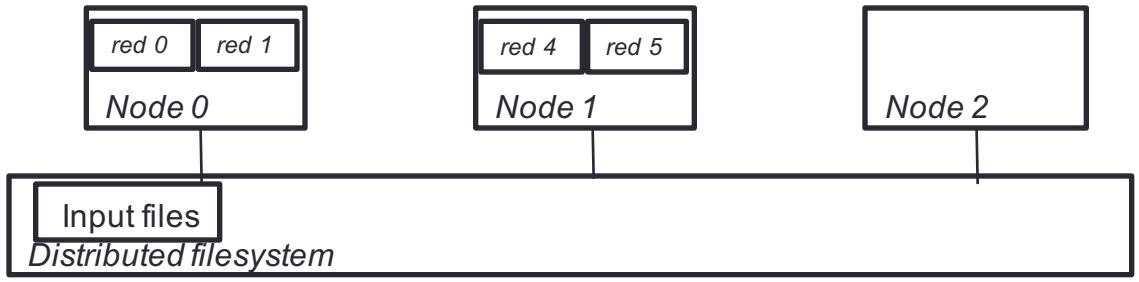
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

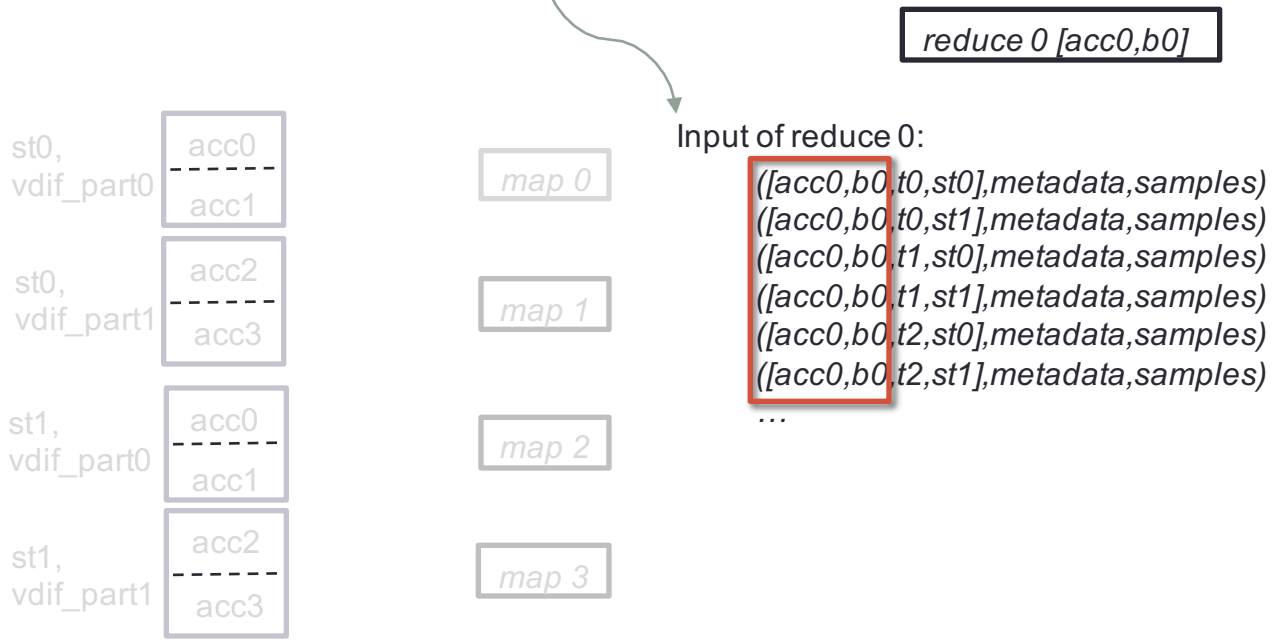
Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

Samples for all stations, acc. period 0, band 0

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.



Mapreduce description

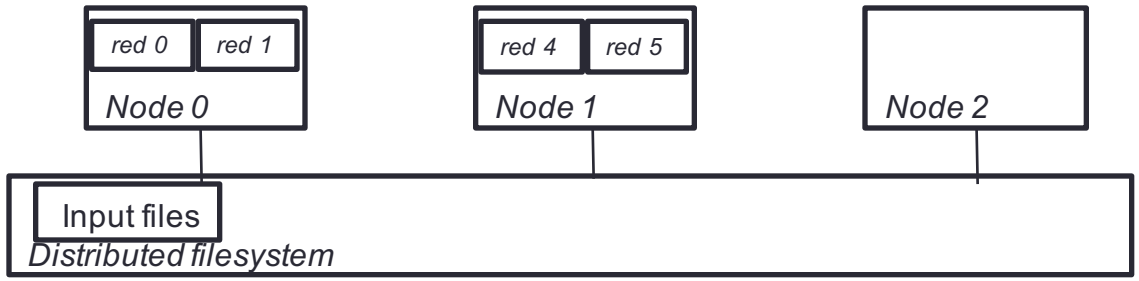
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

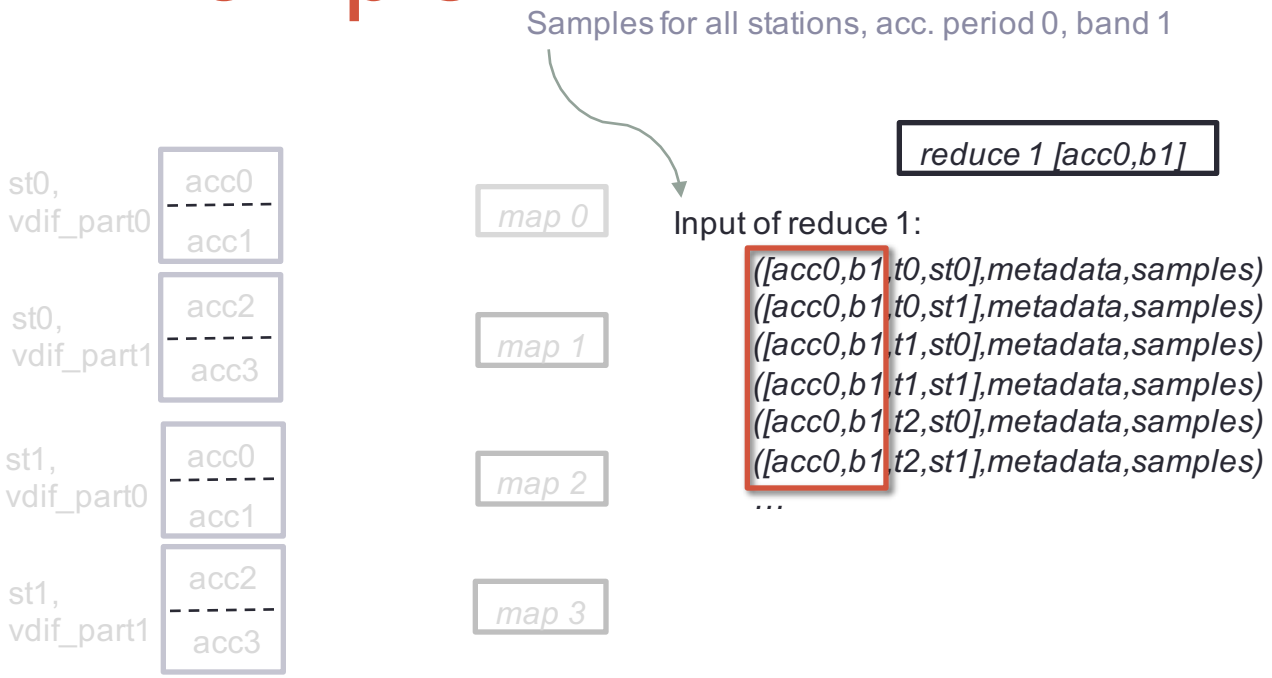
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.



Mapreduce description

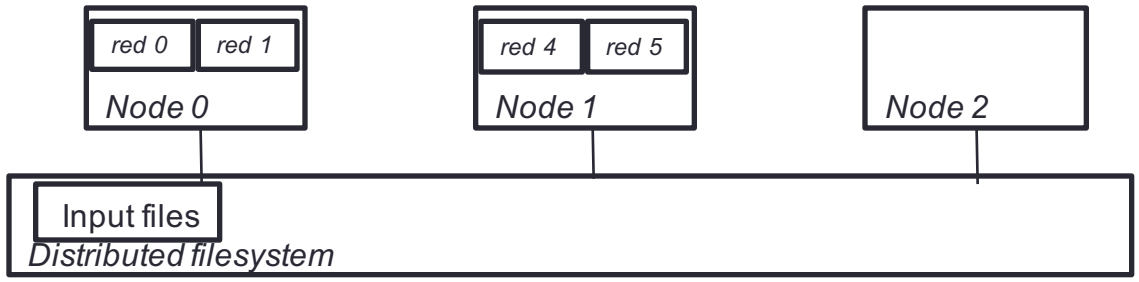
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.

Mapreduce description

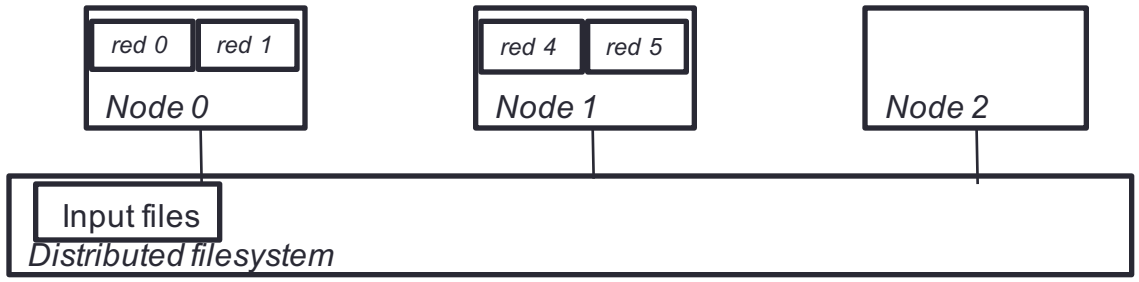
Allocation of tasks and data distribution

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

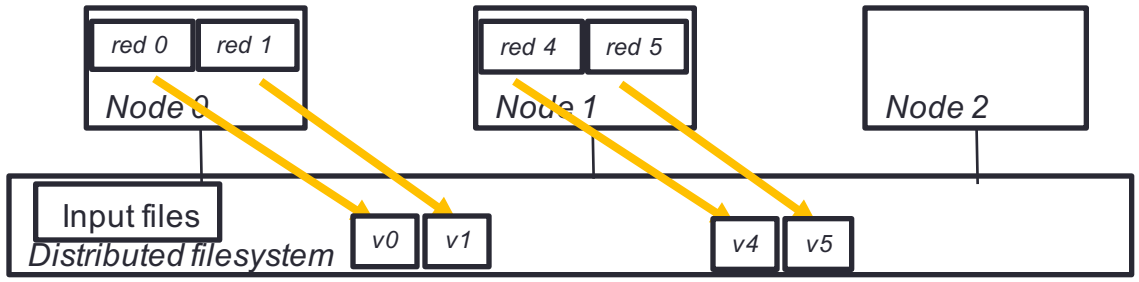
1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.
6. Generation of partial results.

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

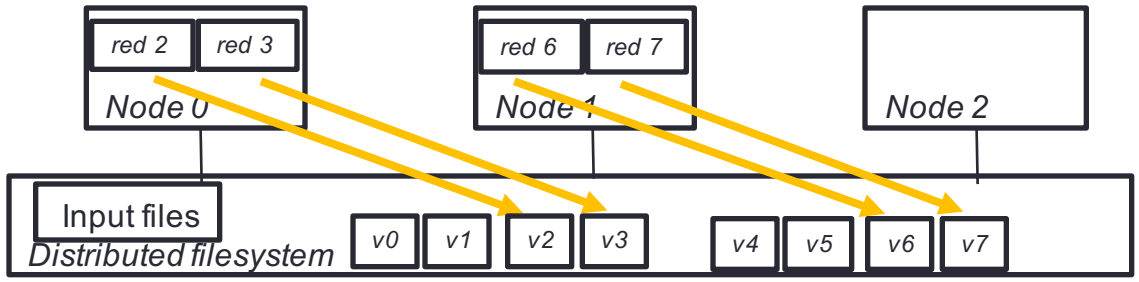
1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.
6. Generation of partial results.

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

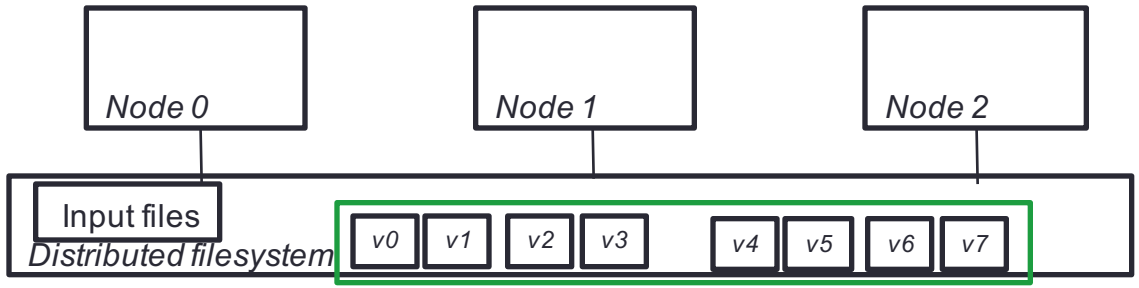
1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.
6. Generation of partial results.
7. Merge results.

MapReduce correlation example:

Number of nodes: 3
Number of cores per node: 2

Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



Correlation Example

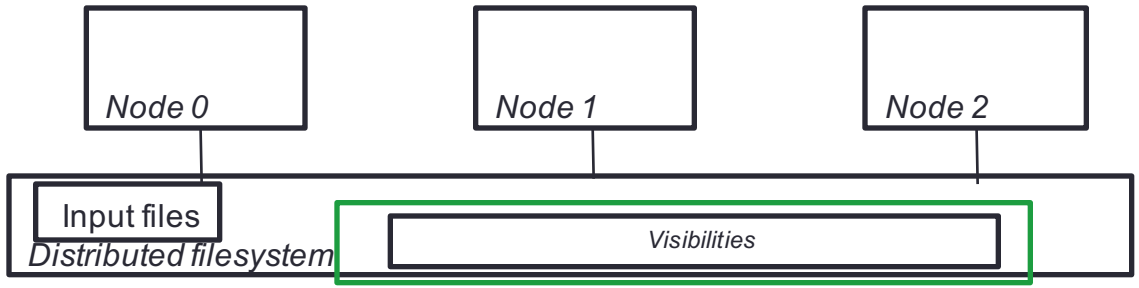
1. Move input data into distributed filesystem.
2. The framework launches one map per input file.
3. Each mapper generates records with keys associated with reducers.
4. The framework distributes and sorts the records.
5. The framework will launch one reducer per partition.
6. Generation of partial results.
7. Merge results.

MapReduce correlation example:

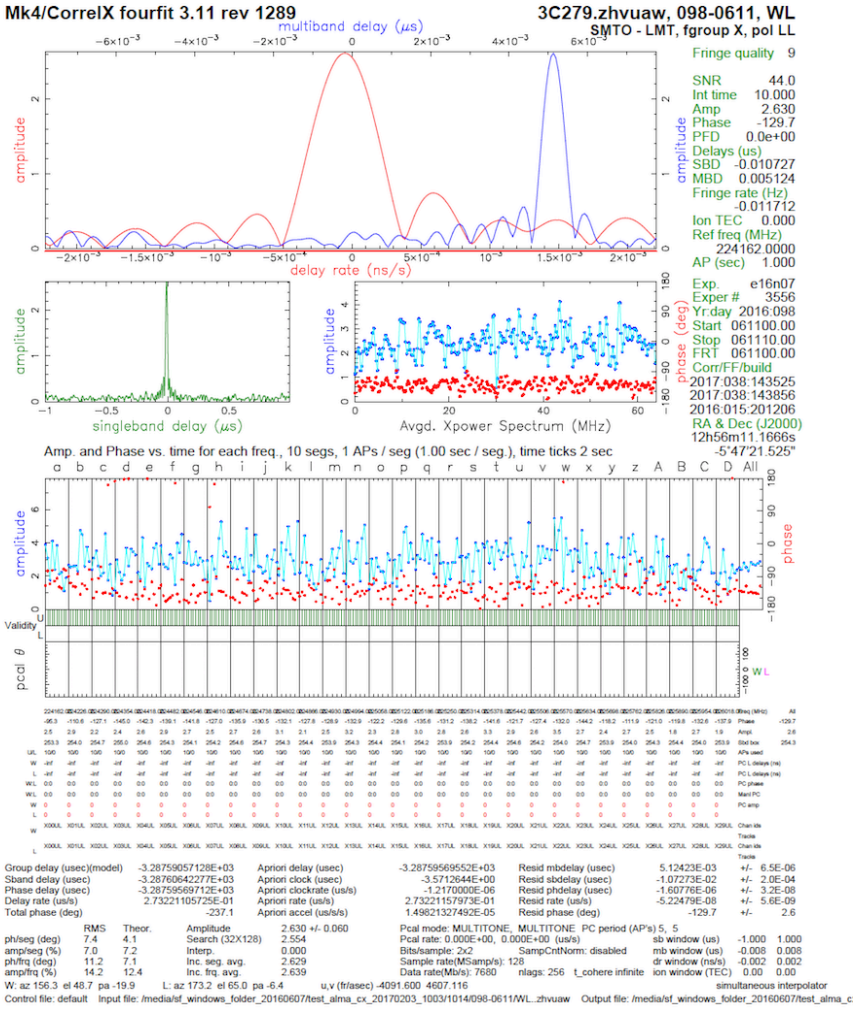
Number of nodes: 3
Number of cores per node: 2

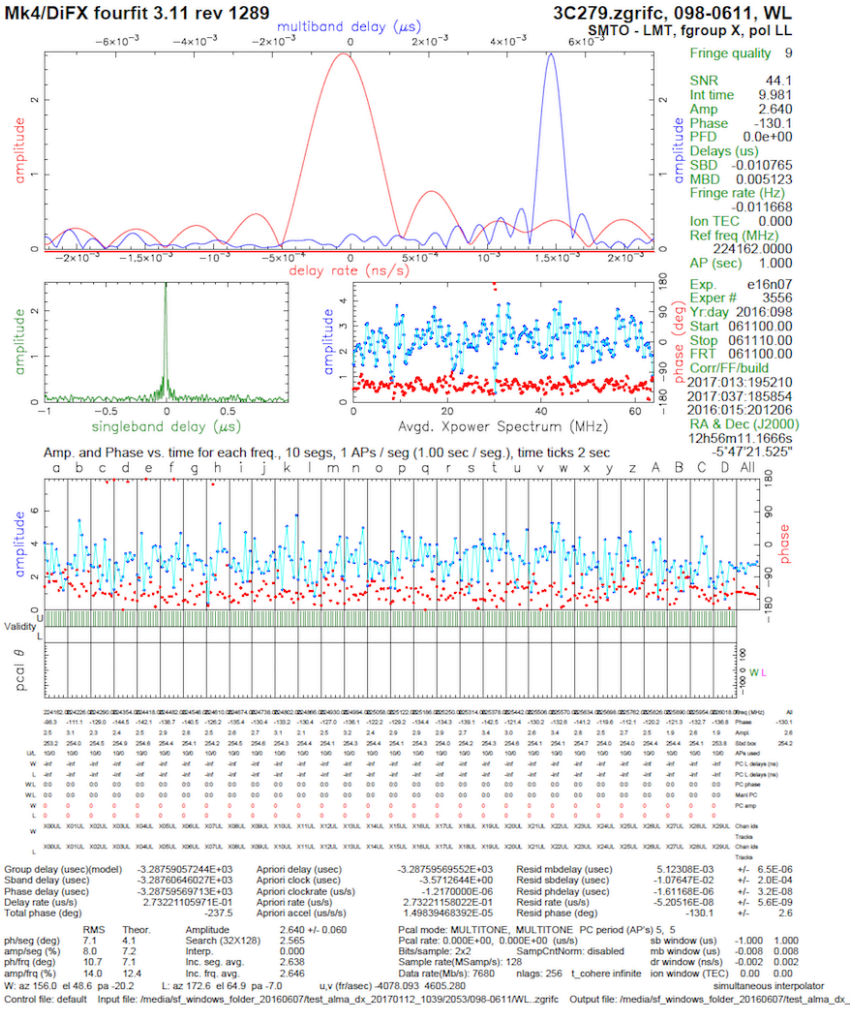
Stations: 2
Input splits: 2 per file.
Acc. Periods: 4.
Bands: 2.

Number of mappers: [sts]*[splits]=4
Number of reducers: [accs]*[bands]=8



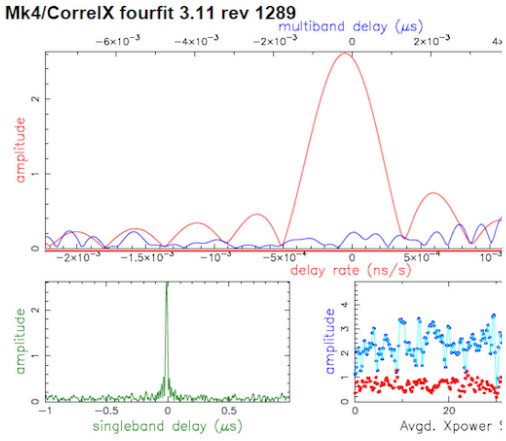
Results - alma_10s_dx_cx_20170207_1059





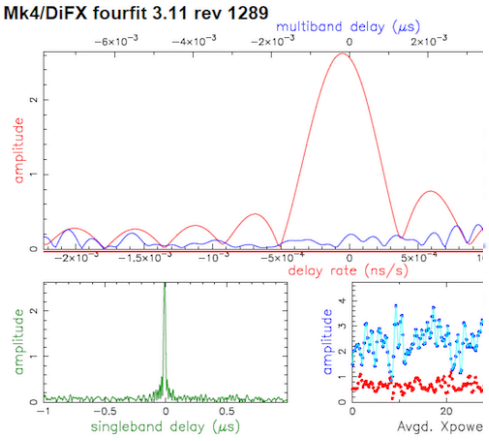
Results - alma_10s_dx_cx_20170207_1059

Fringe quality 9



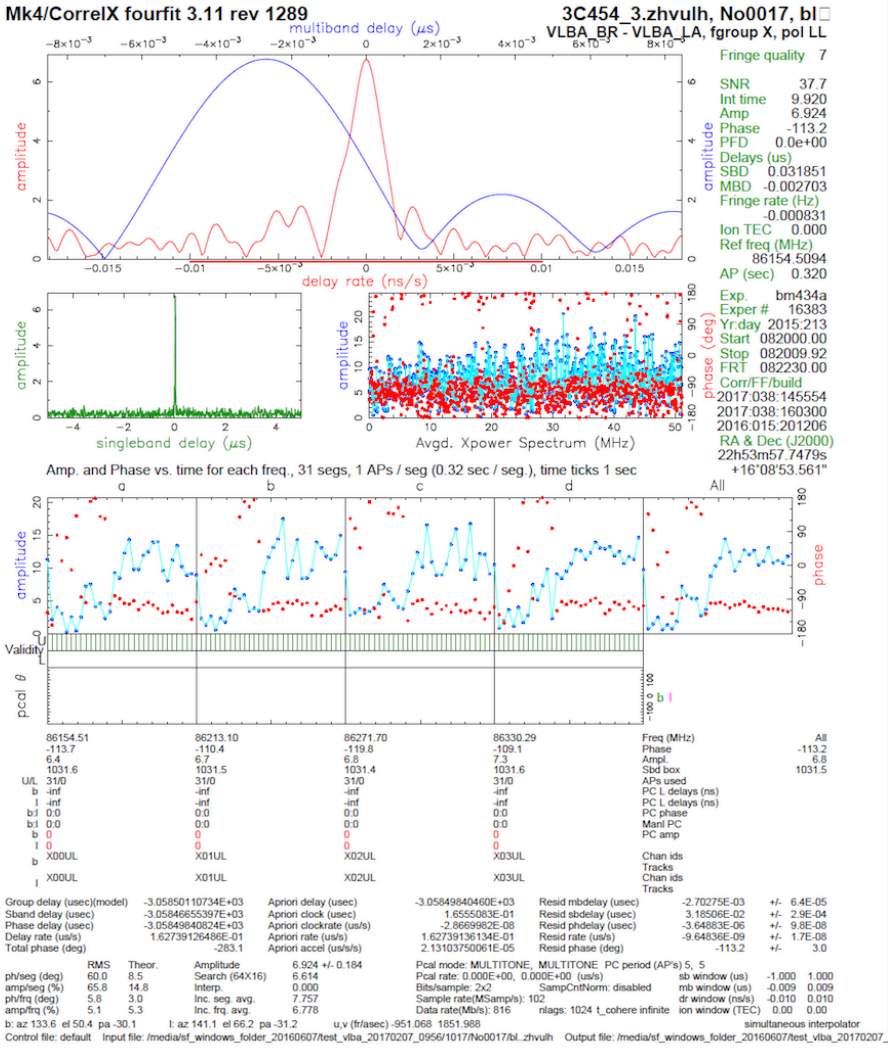
SNR	44.0
Int time	10.000
Amp	2.630
Phase	-129.7
PFD	0.0e+00
Delays (us)	
SBD	-0.010727
MBD	0.005124
Fringe rate (Hz)	
	-0.011712
Ion TEC	0.000
Ref freq (MHz)	
	224162.0000
AP (sec)	1.000
Exp.	e16n07
Exper #	3556
Yr:day	2016:098
Start	061100.00
Stop	061110.00
FRT	061100.00
Corr/FF/build	
	2017:038:143525
	2017:038:143856
	2016:015:201206
RA & Dec (J2000)	
	12h56m11.1666s
	-5°47'21.525"

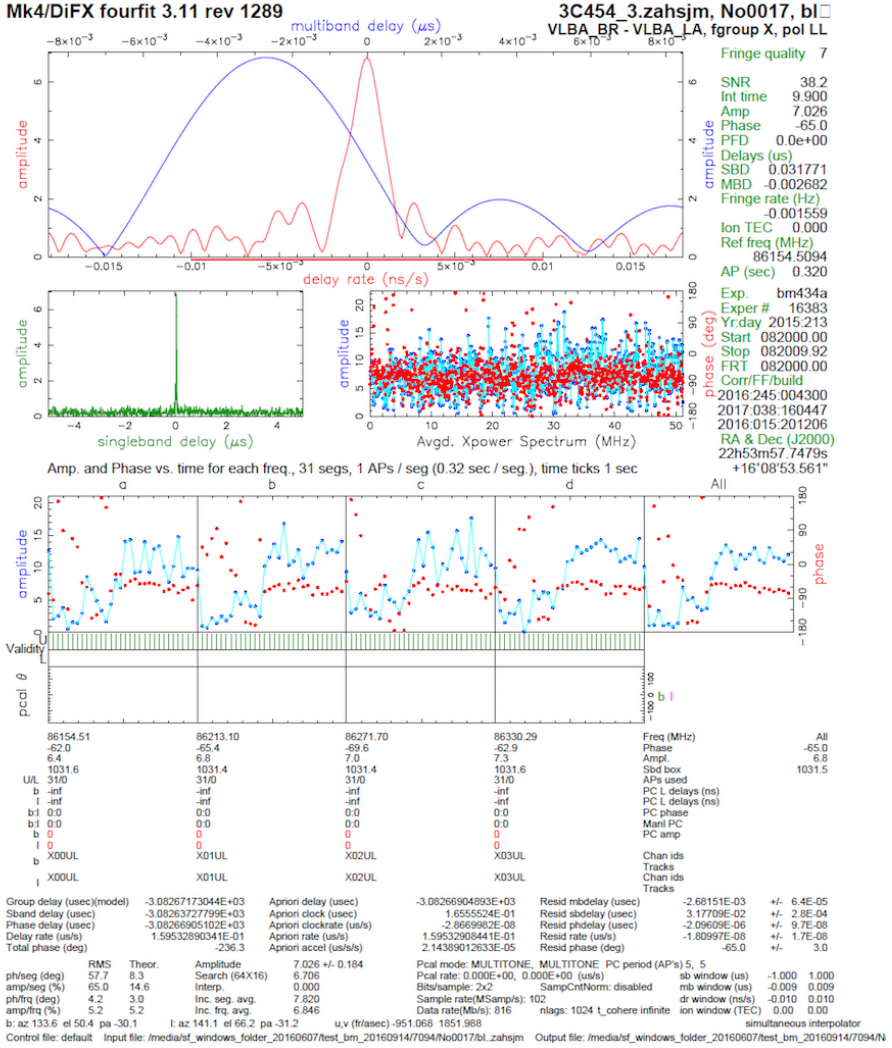
Fringe quality 9



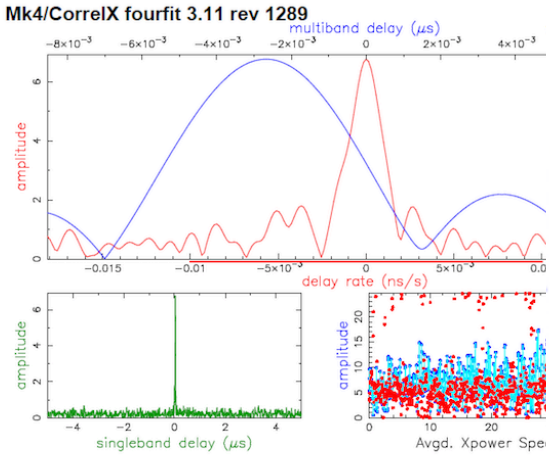
SNR	44.1
Int time	9.981
Amp	2.640
Phase	-130.1
PFD	0.0e+00
Delays (us)	
SBD	-0.010765
MBD	0.005123
Fringe rate (Hz)	
	-0.011668
Ion TEC	0.000
Ref freq (MHz)	
	224162.0000
AP (sec)	1.000
Exp.	e16n07
Exper #	3556
Yr:day	2016:098
Start	061100.00
Stop	061110.00
FRT	061100.00
Corr/FF/build	
	2017:013:195210
	2017:037:185854
	2016:015:201206
RA & Dec (J2000)	
	12h56m11.1666s
	-5°47'21.525"

Results - vlba_10s_dx_cx_20170207_1109



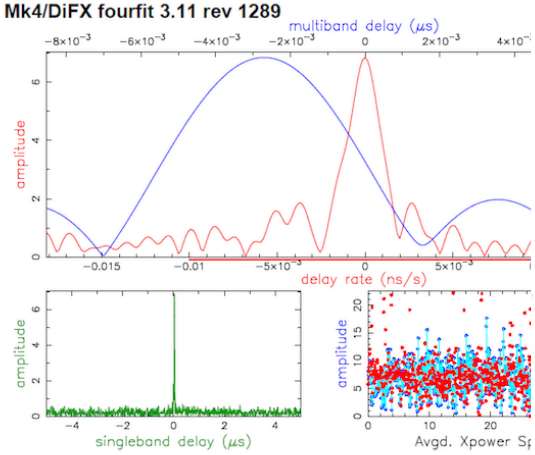


Results - vlba_10s_dx_cx_20170207_1109



Fringe quality 7

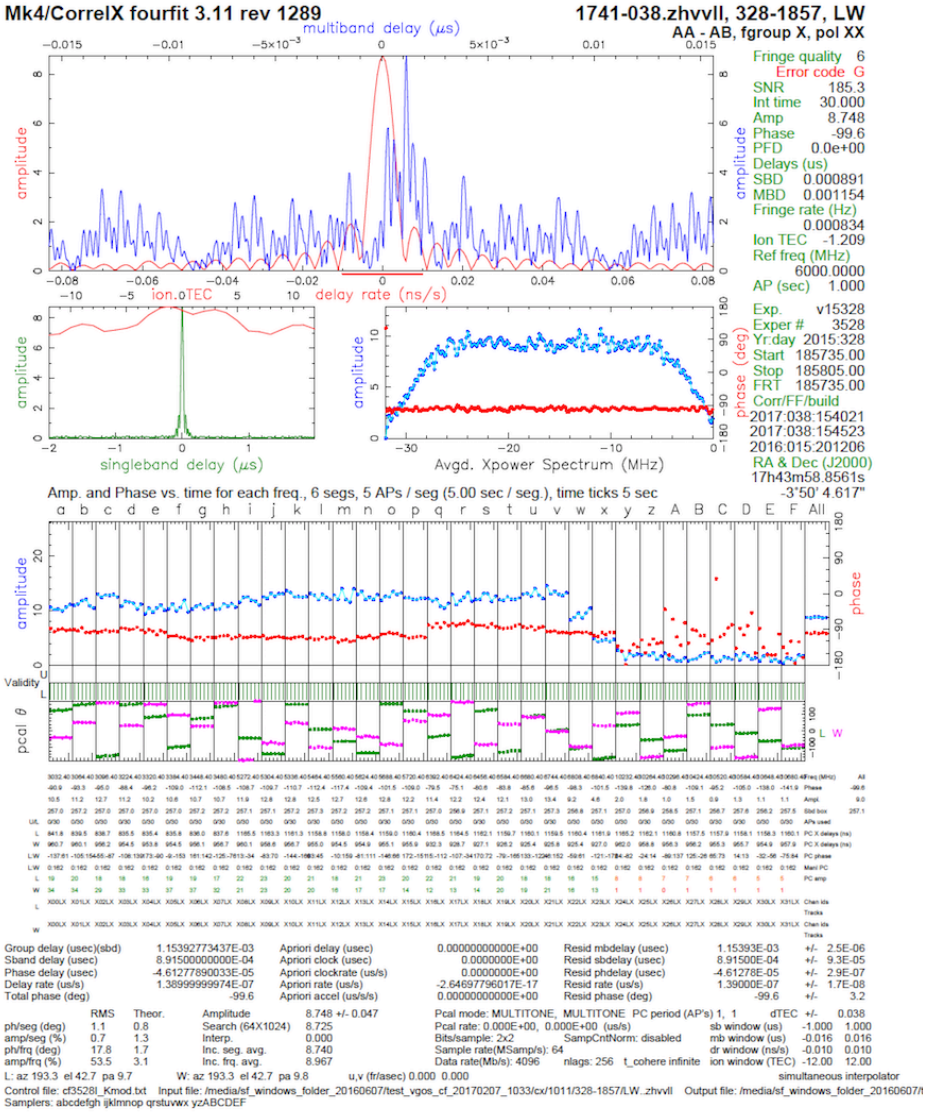
SNR	37.7
Int time	9.920
Amp	6.924
Phase	-113.2
PFD	0.0e+00
Delays (us)	
SBD	0.031851
MBD	-0.002703
Fringe rate (Hz)	
	-0.000831
Ion TEC	0.000
Ref freq (MHz)	
	86154.5094
AP (sec)	0.320
Exp.	bm434a
Exper #	16383
Yr:day	2015:213
Start	082000.00
Stop	082009.92
FRT	082230.00
Corr/FF/build	
	2017:038:145554
	2017:038:160300
	2016:015:201206
RA & Dec (J2000)	
	22h53m57.7479s
	+16°08'53.561"

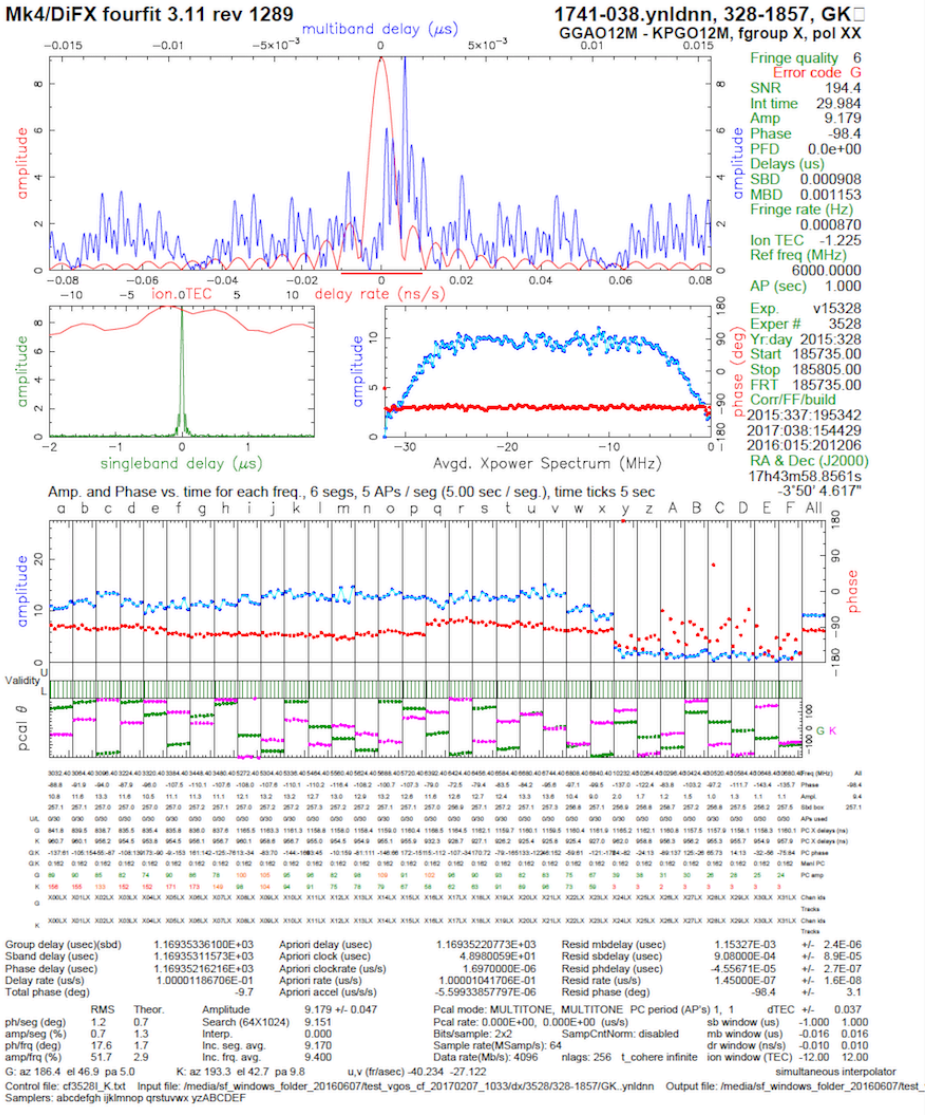


Fringe quality 7

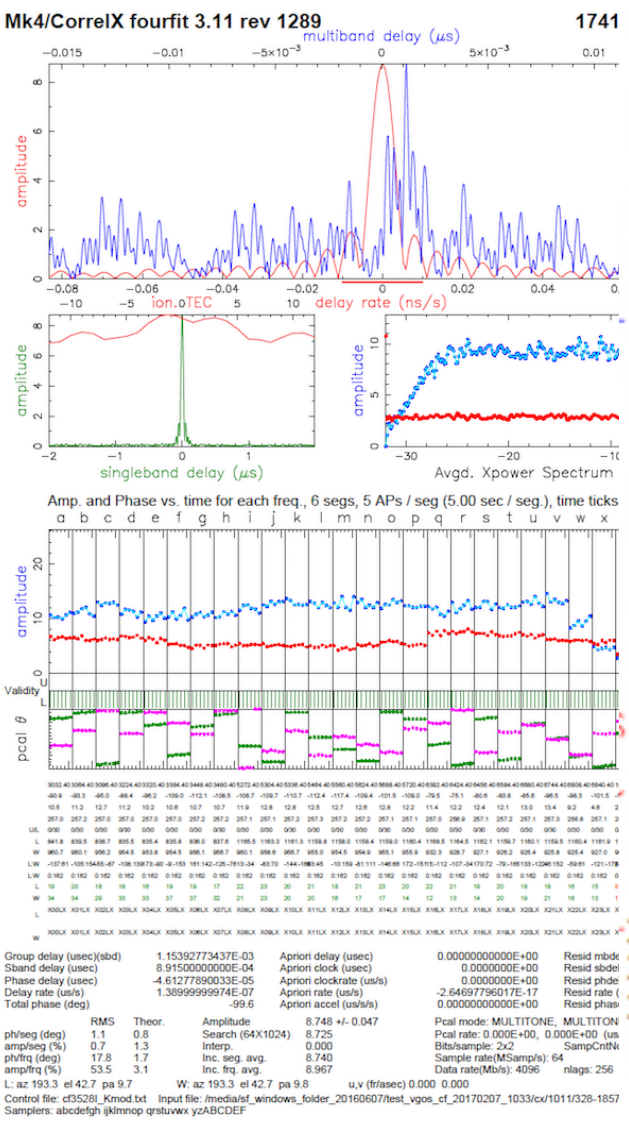
SNR	38.2
Int time	9.900
Amp	7.026
Phase	-65.0
PFD	0.0e+00
Delays (us)	
SBD	0.031771
MBD	-0.002682
Fringe rate (Hz)	
	-0.001559
Ion TEC	0.000
Ref freq (MHz)	
	86154.5094
AP (sec)	0.320
Exp.	bm434a
Exper #	16383
Yr:day	2015:213
Start	082000.00
Stop	082009.92
FRT	082000.00
Corr/FF/build	
	2016:245:004300
	2017:038:160447
	2016:015:201206
RA & Dec (J2000)	
	22h53m57.7479s
	+16°08'53.561"

Results - vgos_cf_dx_cx_20170207_1056



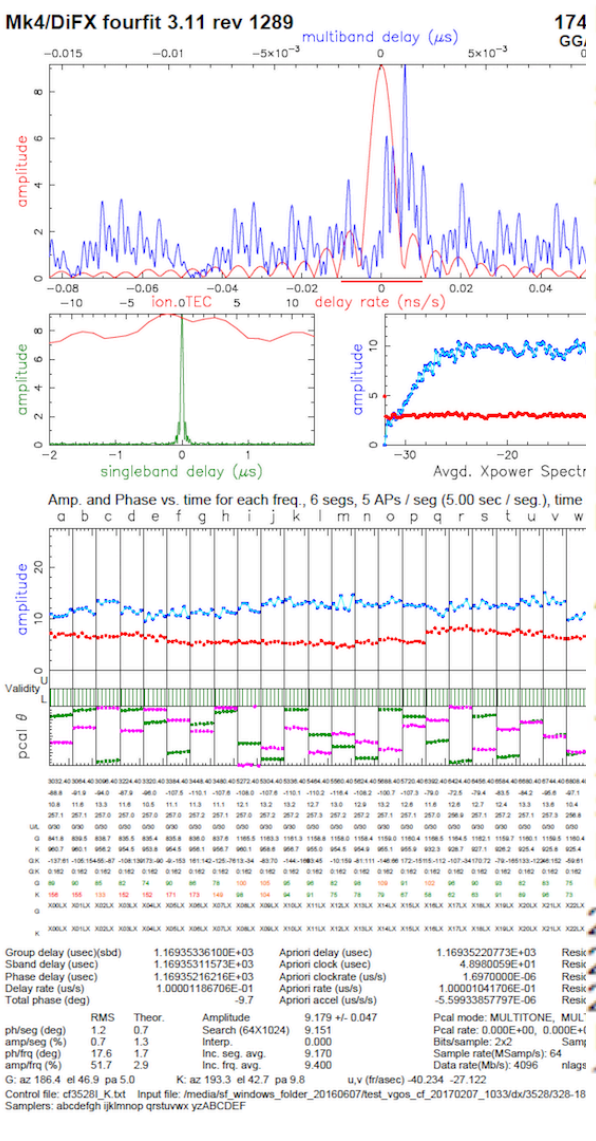


Results - vgos_cf_dx_cx_20170207_1056



Fringe quality 6
Error code G
SNR 185.3
Int time 30.000
Amp 8.748
Phase -99.6
PFD 0.0e+00
Delays (us)
SBD 0.000891
MBD 0.001154
Fringe rate (Hz)
0.000834
Ion TEC -1.209
Ref freq (MHz)
6000.0000
AP (sec) 1.000

Exp. v15328
Exper # 3528
Yr:day 2015:328
Start 185735.00
Stop 185805.00
FRT 185735.00
Corr/FF/build
2017:038:154021
2017:038:154523
2016:015:201206
RA & Dec (J2000)
17h43m58.8561s
-3°50' 4.617"

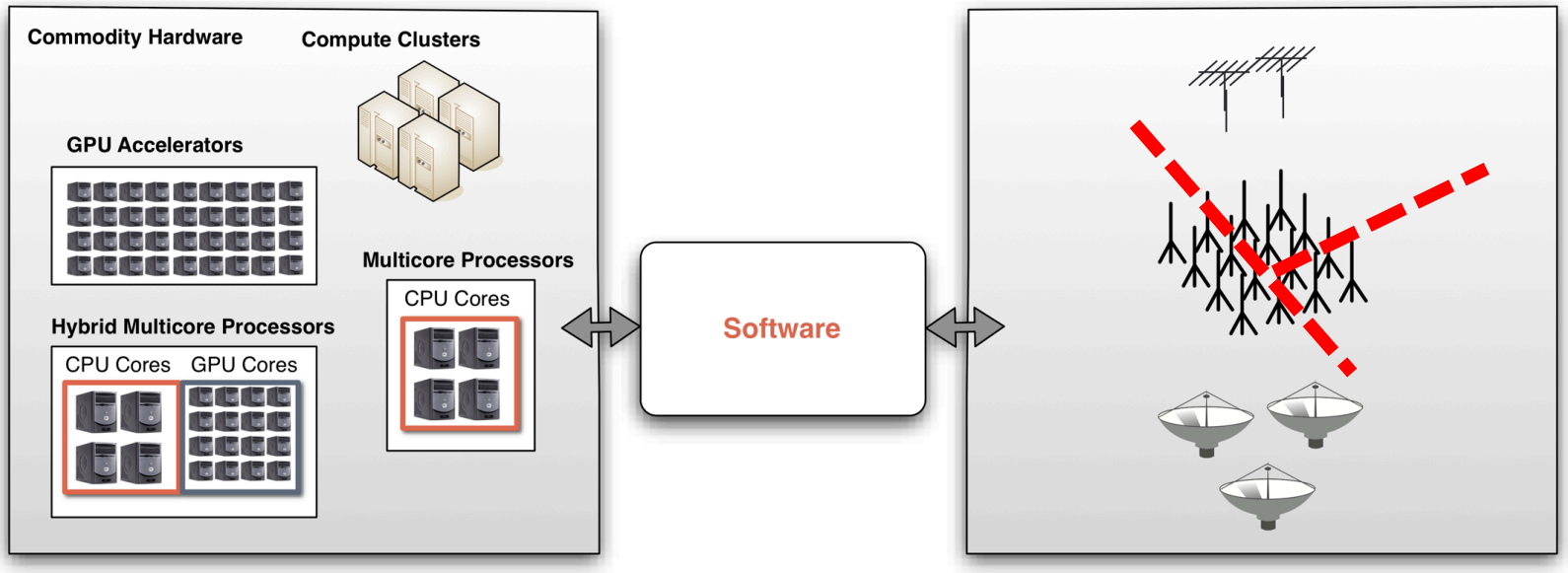


Fringe quality 6
Error code G
SNR 194.4
Int time 29.984
Amp 9.179
Phase -98.4
PFD 0.0e+00
Delays (us)
SBD 0.000908
MBD 0.001153
Fringe rate (Hz)
0.000870
Ion TEC -1.225
Ref freq (MHz)
6000.0000
AP (sec) 1.000

Exp. v15328
Exper # 3528
Yr:day 2015:328
Start 185735.00
Stop 185805.00
FRT 185735.00
Corr/FF/build
2015:337:195342
2017:038:154429
2016:015:201206
RA & Dec (J2000)
17h43m58.8561s
-3°50' 4.617"

Perspectives

Chunk model enables partitioning of large instrument arrays to work in different modes at the same time.



Chunk model enables execution on heterogeneous hardware. Different hardware → different reducers for chunks. Will use machine learning for performance tuning of chunk execution on each platform.

Conclusion

- CorrelX
  - **Flexibility, scalability, simplicity**
  - Proof-of-concept functionality for VGOS, ALMA, VLBA correlations
    - VGOS: LSB 2-bit complex, phase calibration
    - EHT: USB 2-bit real, zoom bands
    - Integrated in state-of-the-art processing chain
  - Plugins: Opportunity for research and teaching

→Easily prototype new algorithms, mapper-reducers, execute on real data sets

Acknowledgements

- We would like to thank the MIT Haystack Observatory staff, in particular A. **Rogers** and R. **Capallo** for invaluable help during developing and testing, and J. **Barrett**, G. **Crew**, M. **Gowanlock**, J. **Li**, G. **Rongier**, C. **Rude**, and M. **Titus** for generous feedback. We acknowledge partial support from the **NASA Space Geodesy Program** and the **NSF** for supporting the Massachusetts Green High Performance Computing Center (**MGHPCC**) for our experiments in Holyoke, Massachusetts.

Thanks!

Questions? Comments? Want to contribute?

correlX@haystack.mit.edu