Cloud Computing Technologies and Applications

TVB ATV CABLE TV Phoenix TV

What we have done in the first semester

Technologies & Applications

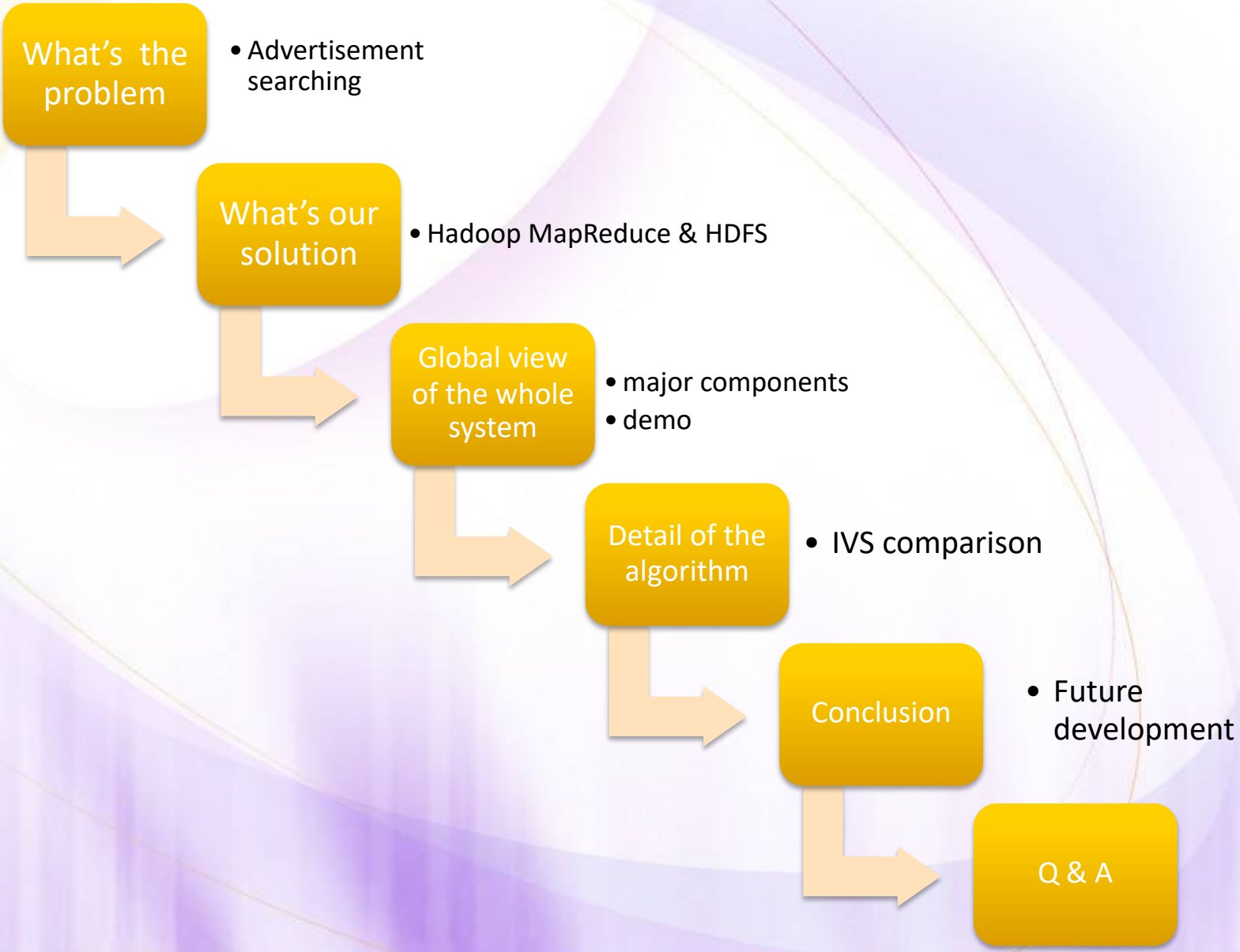
Research on Private cloud : Eucalyptus
Research on Hadoop MapReduce & HDFS

How about this semester

Technologies & Applications

Use the technologies to solve real problems

Contents


What's the problem
• Advertisement searching
What's our solution
• Hadoop MapReduce & HDFS
Global view of the whole system
• major components
• demo
Detail of the algorithm
• IVS comparison
Conclusion
• Future development
Q & A

What's the problem?

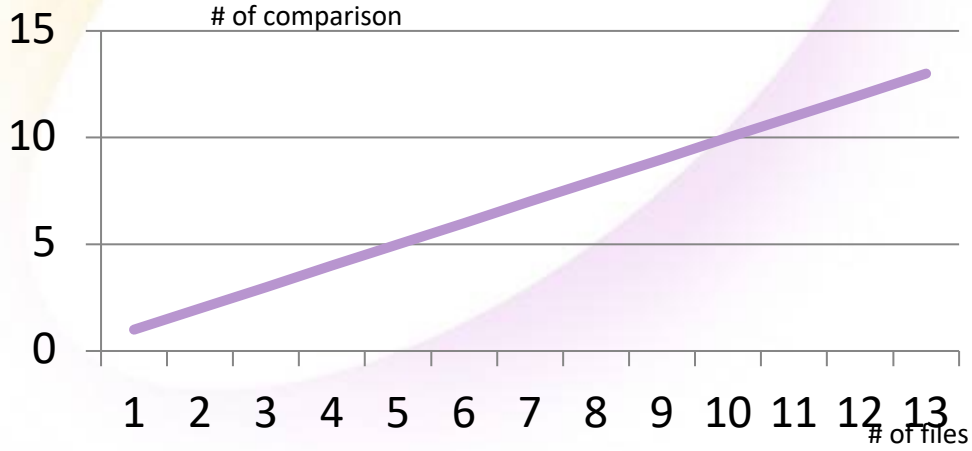
Find out how many, at what time the advertisements being broadcast

Record the videos , 1 hour as a file

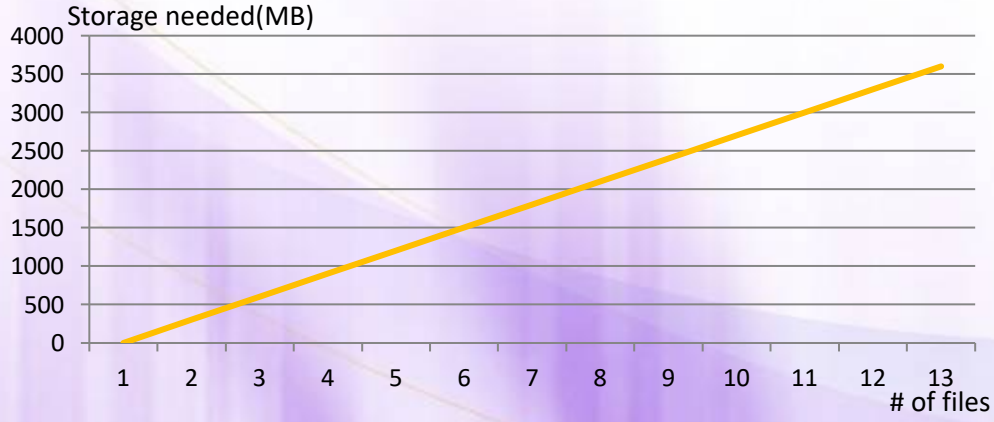
Compare the latest one on each of the existing one

What's the problem?

The # of comparison keeps linearly increasing with time



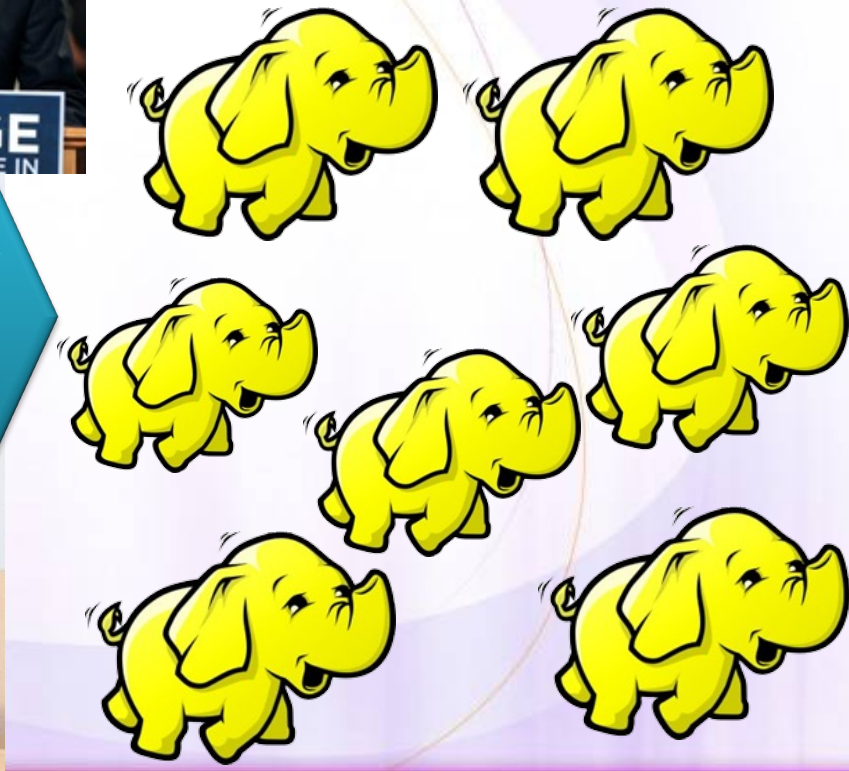
The storage requirement keeps linearly increasing with time



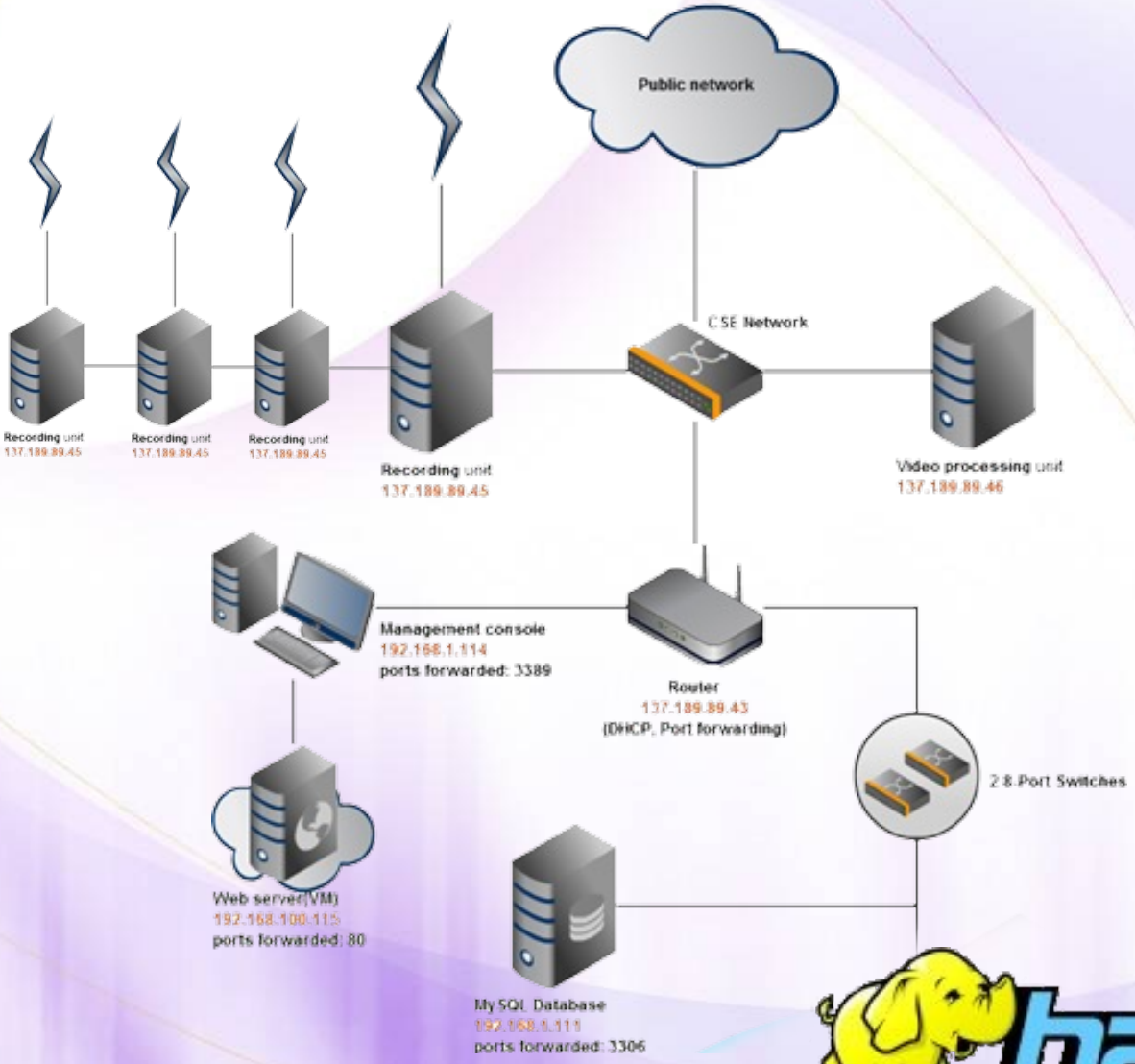
* Considering one channel

The system needs to Scale Up!

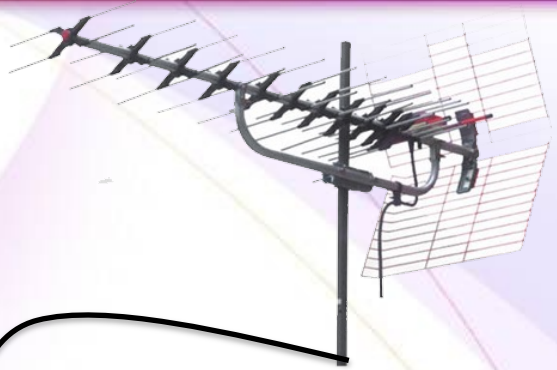
- **Computational Power**
- **Storage**



Our solution



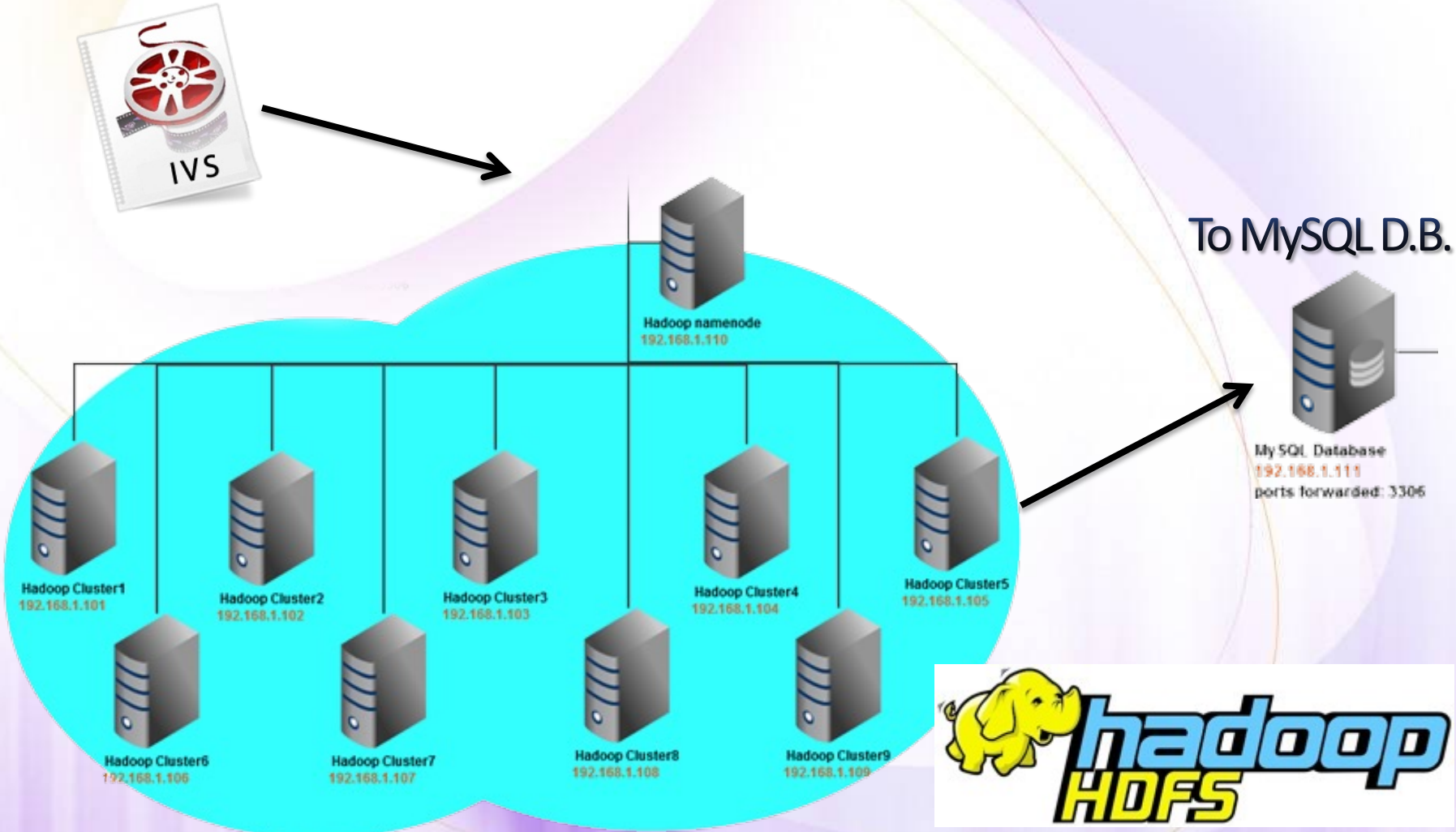
Recording units



Generate a MPEG file in each hour

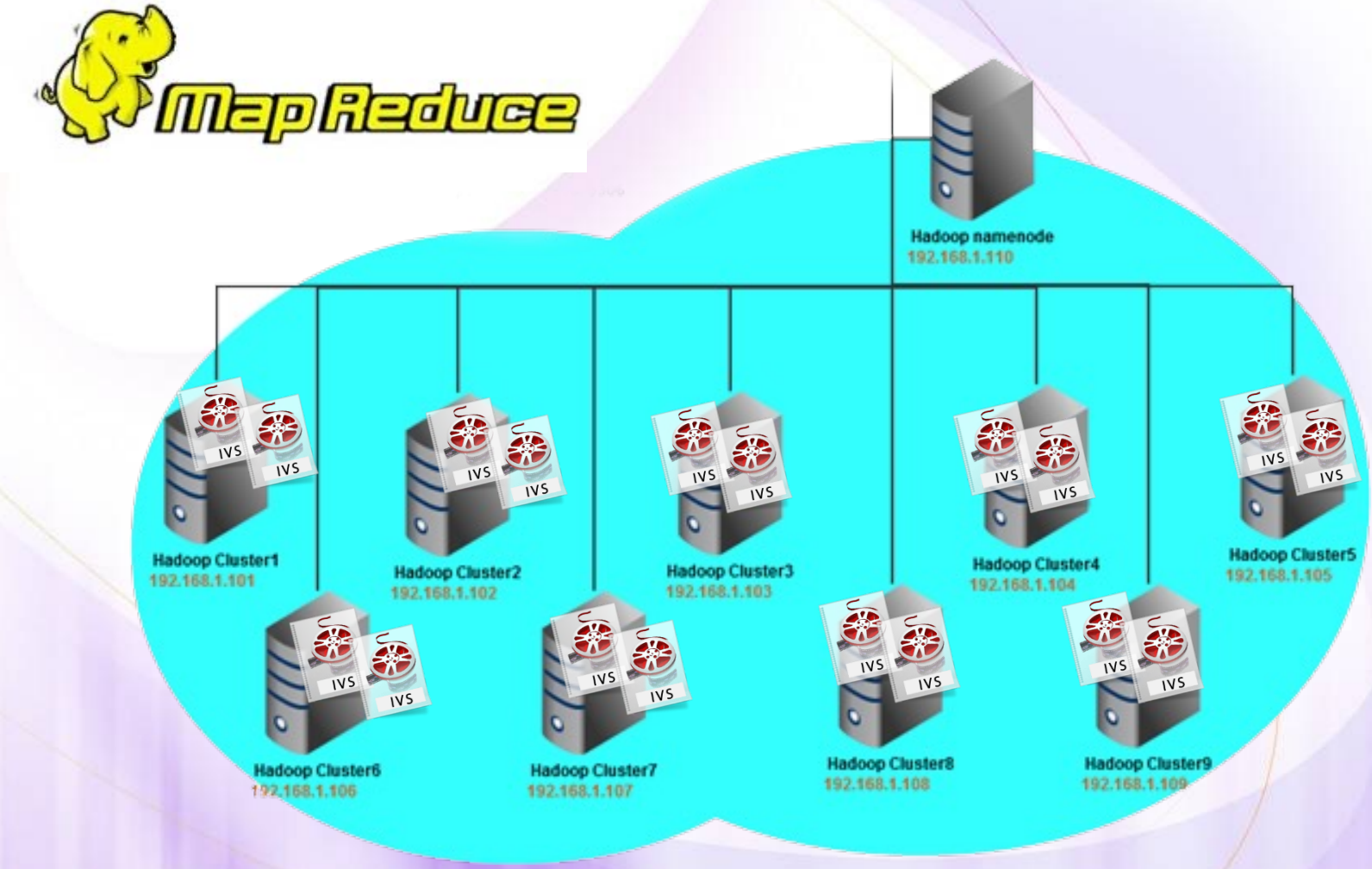
Video processing unit

IVS comparison unit

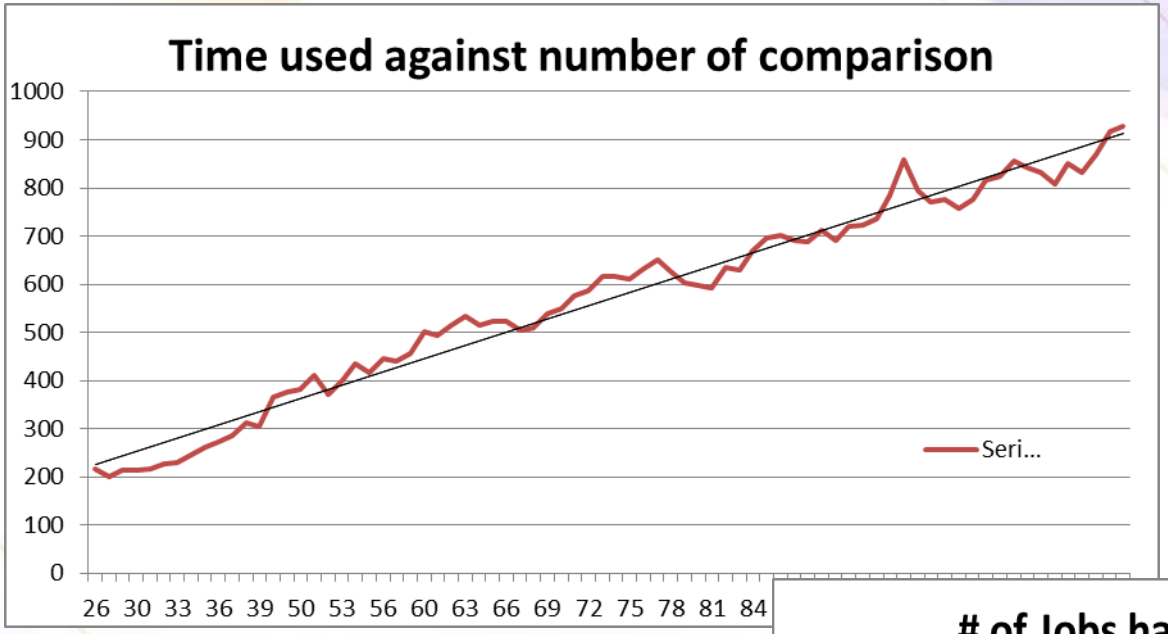


Hadoop clusters

IVS comparison unit



IVS comparison unit



IVS comparison unit

task list for job_201104182255_0001

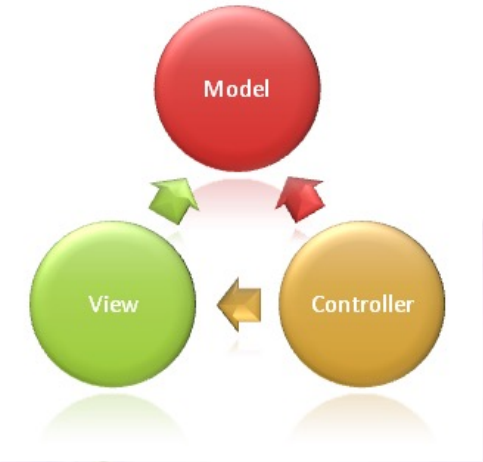
Task Id	Start Time	Finish Time	Error
task_201104182255_0001_m_000002	18/04 22:50:31	18/04 22:51:04 (32sec)	
task_201104182255_0001_m_000001	18/04 22:02:50	18/04 22:03:16 (25sec)	
task_201104182255_0001_m_000000	18/04 22:34:16	18/04 22:34:53 (37sec)	
task_201104182255_0001_m_000006	18/04 22:48:18	18/04 22:48:46 (28sec)	
task_201104182255_0001_m_000005	18/04 22:56:07	18/04 22:56:43 (35sec)	
task_201104182255_0001_m_000004	18/04 22:47:21	18/04 22:48:01 (39sec)	
task_201104182255_0001_m_000003	18/04 22:40:30	18/04 22:41:07 (37sec)	
task_201104182255_0001_m_000010	18/04 22:28:59	18/04 22:29:41 (41sec)	
task_201104182255_0001_m_000009	18/04 22:57:04	18/04 22	
task_201104182255_0001_m_000008	18/04 22:40:50	18/04 22	
task_201104182255_0001_m_000007	18/04 22:28:36	18/04 22	
task_201104182255_0001_m_000011	18/04 22:03:17	18/04 22	
task_201104182255_0001_m_000012	18/04 22:57:32	18/04 22	

task_201104182255_0002_m_000023	18/04 22:40:47	18/04 22:40:59 (12sec)	Another (possibly speculative) attempt already SUCCEEDED
task_201104182255_0002_m_000022	18/04 22:34:02	18/04 22:35:05 (1mins, 2sec)	
task_201104182255_0002_m_000022	18/04 22:54:44	18/04 22:54:53 (9sec)	Another (possibly speculative) attempt already SUCCEEDED
task_201104182255_0002_m_000025	18/04 22:39:49	18/04 22:40:31 (41sec)	
task_201104182255_0002_m_000024	18/04 22:53:46	18/04 22:54:14 (28sec)	
task_201104182255_0002_m_000013	18/04 23:00:28	18/04 23:01:39 (1mins, 11sec)	
task_201104182255_0002_m_000012	18/04 22:46:08	18/04 22:46:39 (30sec)	Another (possibly speculative) attempt already SUCCEEDED
task_201104182255_0002_m_000012	18/04 22:54:52	18/04 22:56:31 (1mins, 39sec)	
task_201104182255_0002_m_000015	18/04 23:01:28	18/04 23:02:30 (1mins, 1sec)	
task_201104182255_0002_m_000014	18/04 22:38:43	18/04 22:39:48 (1mins, 5sec)	

I/O overhead

The Front end : Web Application

Shows the result that our system captured

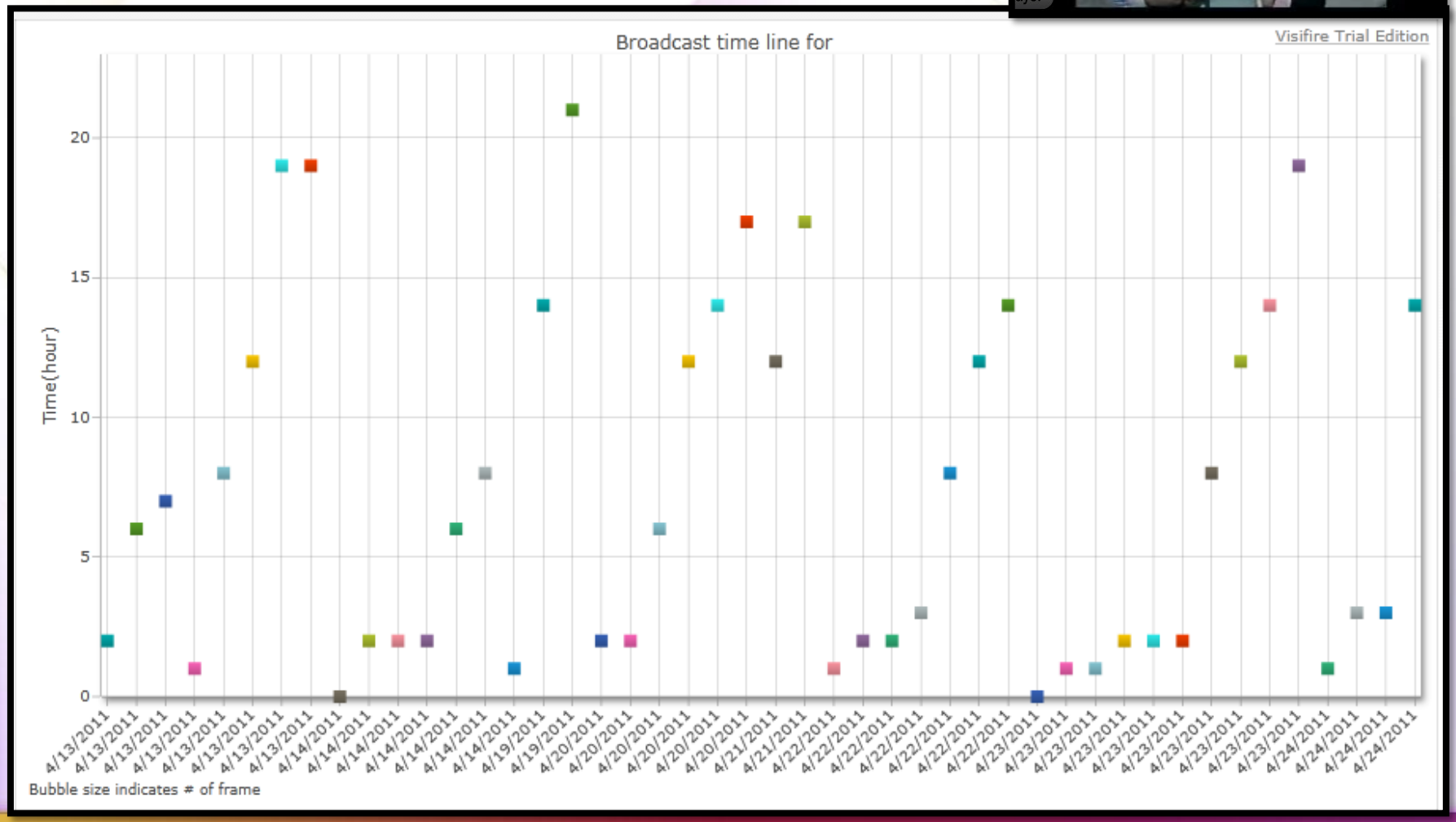


Asp.net
MVC 2.0

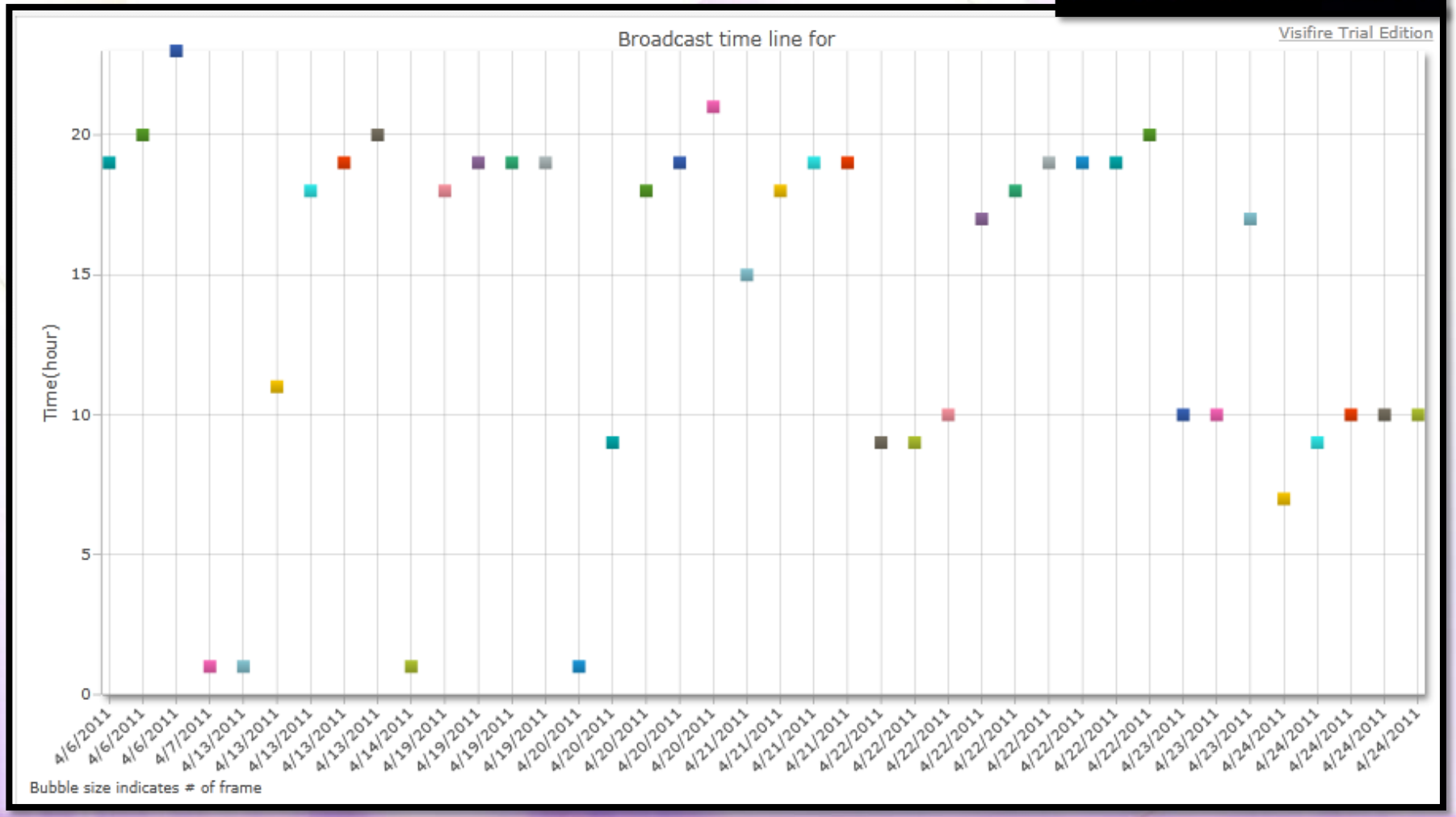
Demo. on IVS Web Application

http://ivs.vvfun.com

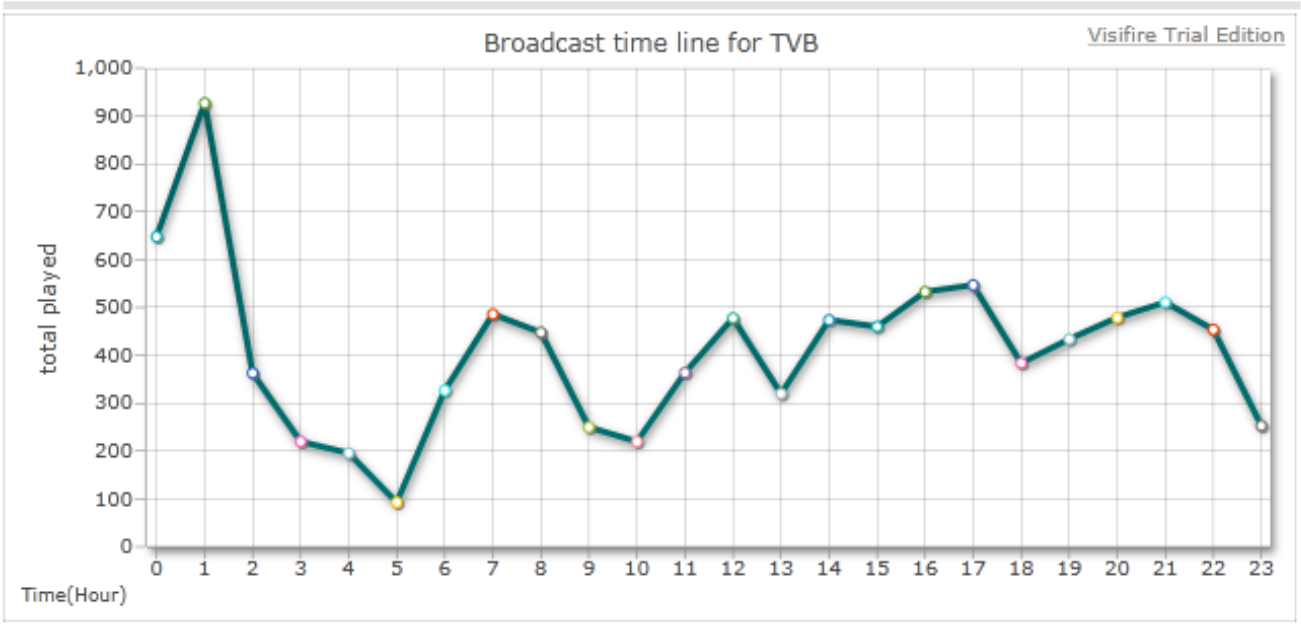
Some captured data

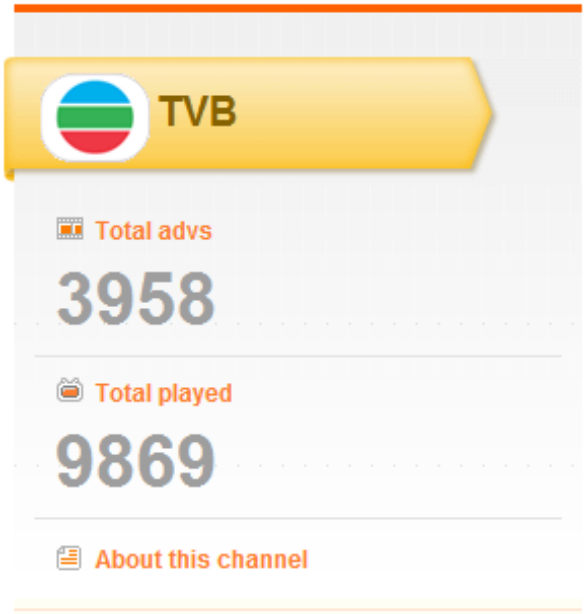


Some captured data

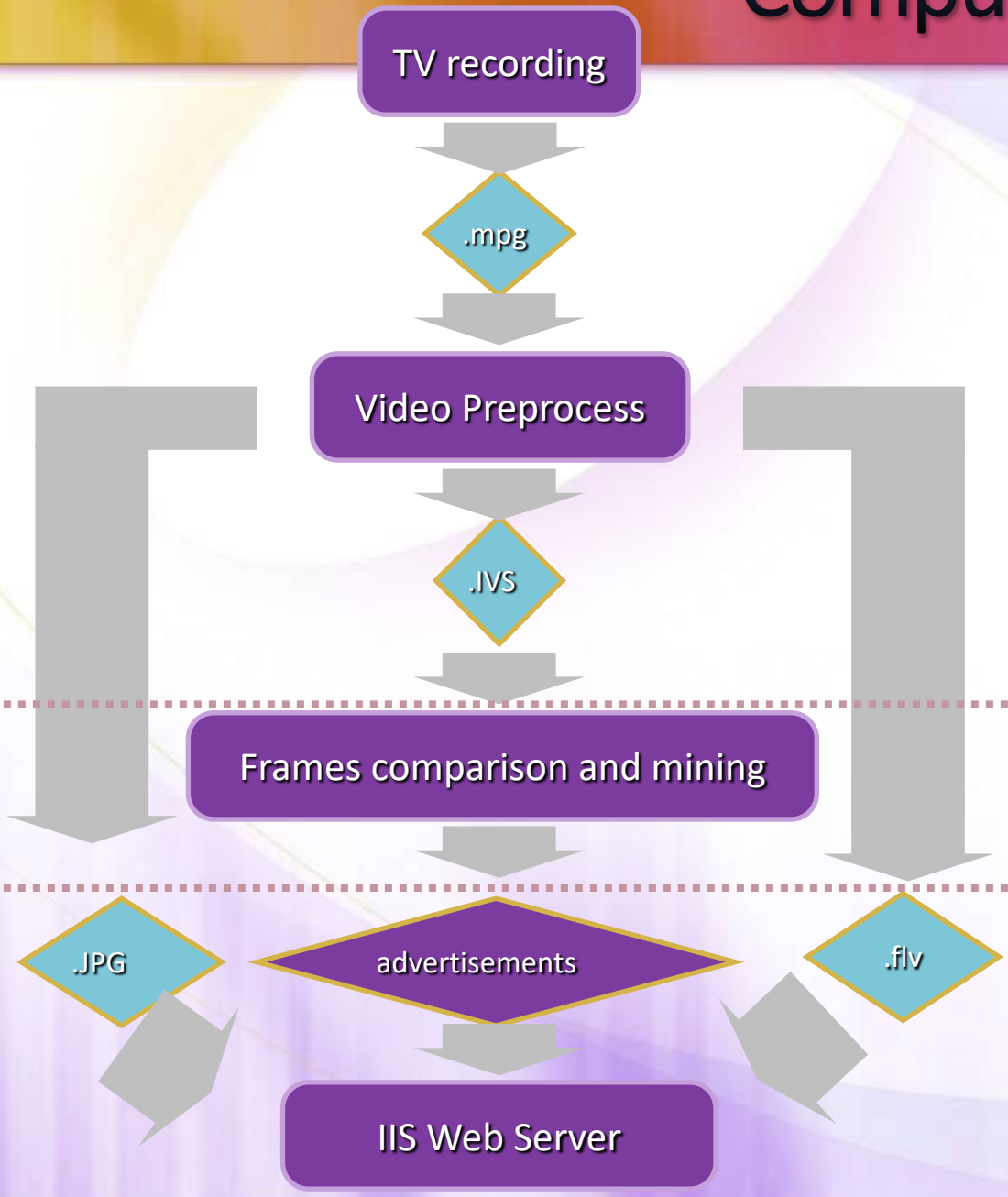


Some captured data





Computational Complexity



O(1)

O(n) Hadoop used here

O(1)

How to find out the advertisements

THE  OF I.V.S.

IVS

- **I**nvariant **V**ideo **S**ignature
- Developed by VIEW lab
- Convert each frame to a vector with 1024 attributes
- Save storage
- Save computation power

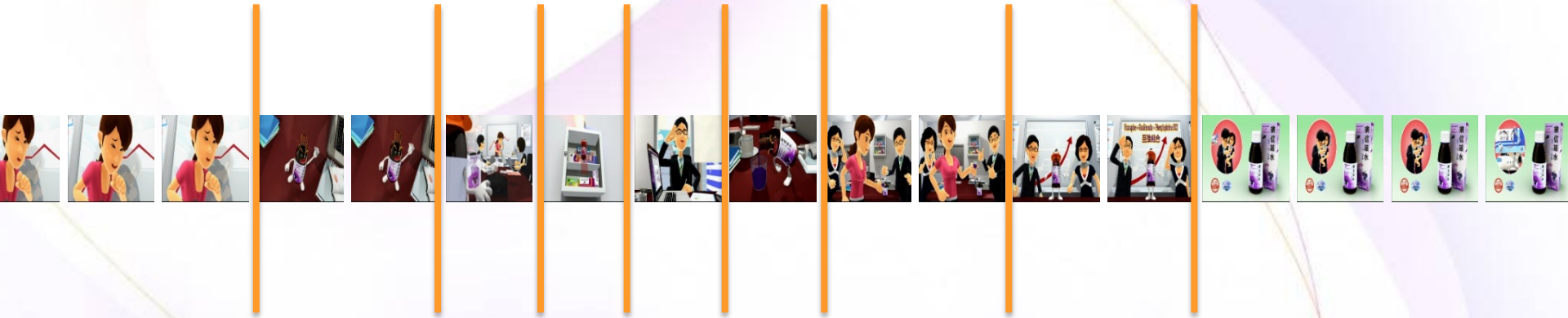
Brief idea

- Find the longest common subsequence
- $O(n^2)$
- n= 60*60*25=90000
- n^2 =8.1e9
  - e.g. compare one hour with another one hour
  - ~ 1 hour/ comparison

- Do something to reduce the n

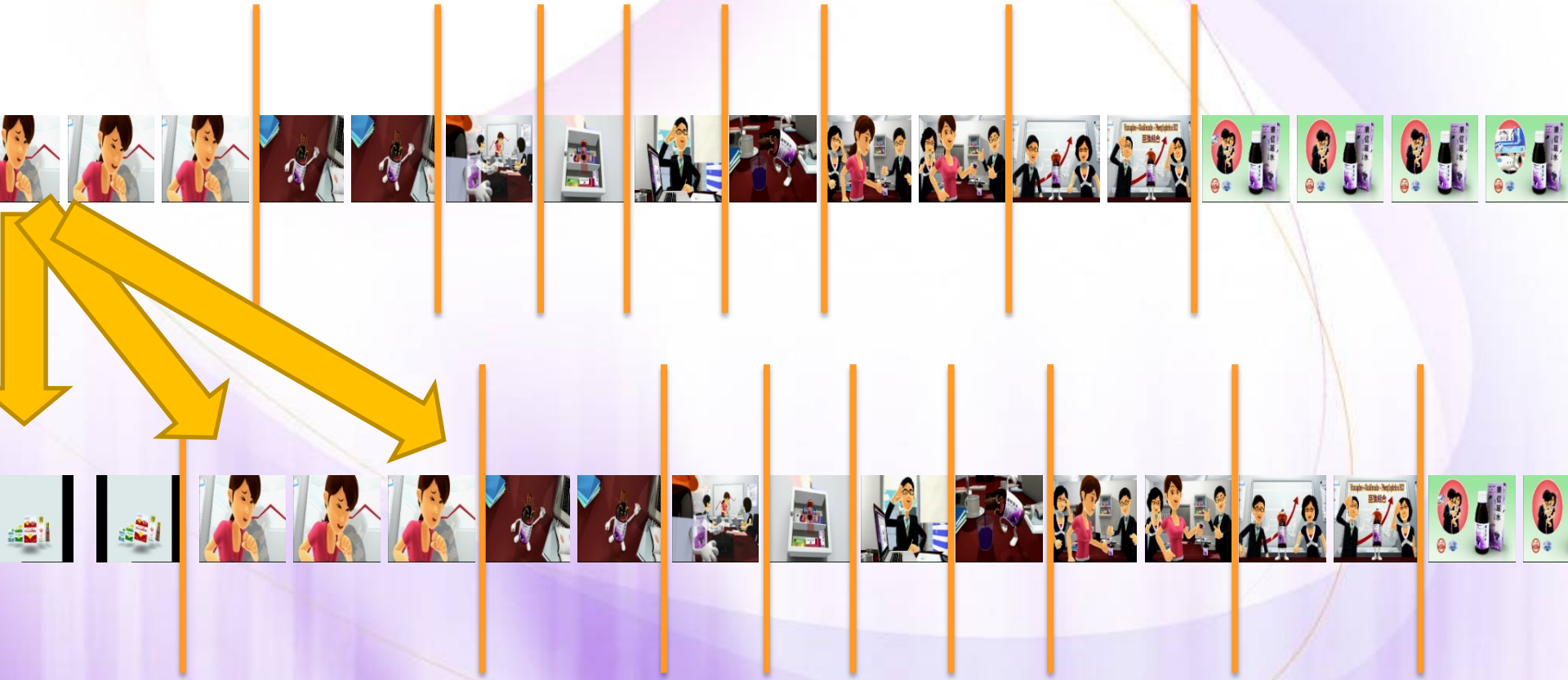
How to find a common subsequence

- Step 1 : Group similar frames
  - ~ 750 frames become ~10-20 groups(segments)



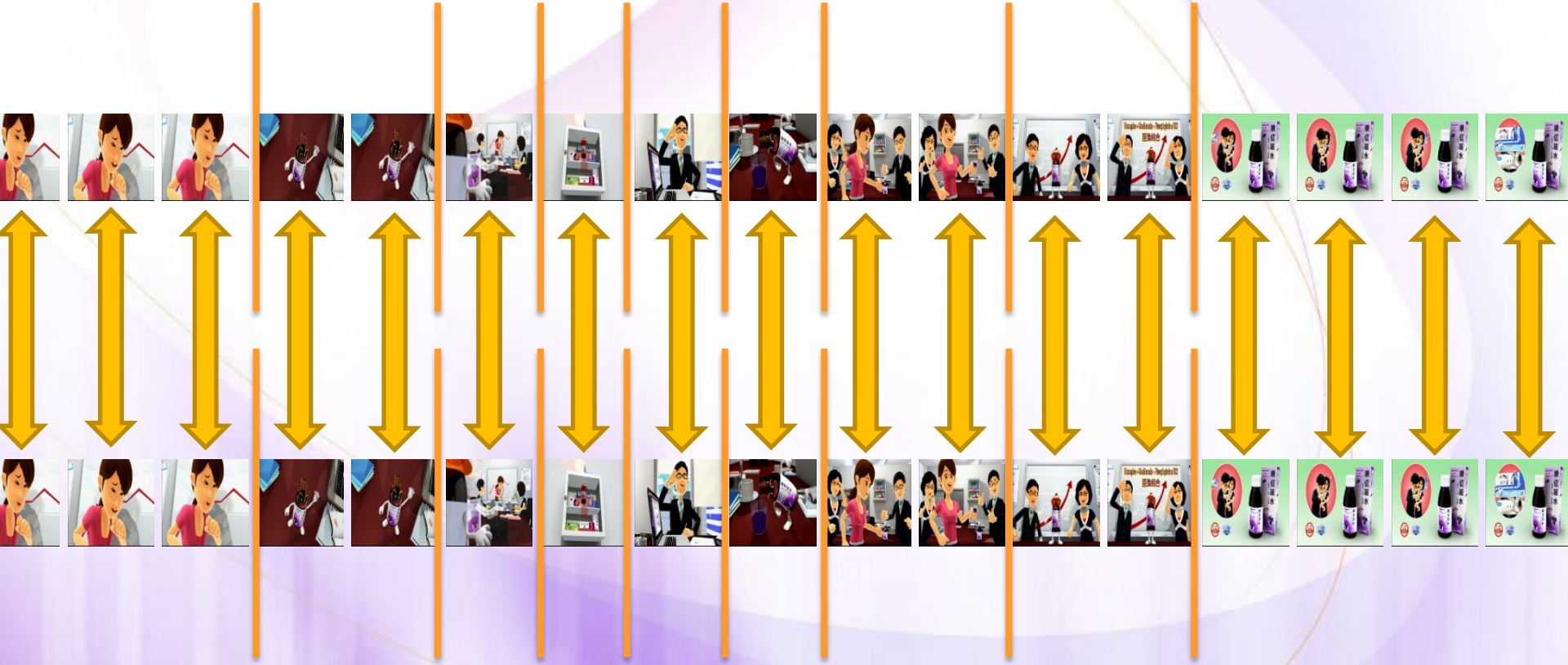
How to find a common subsequence

- Step 2: Compare the first frame of each group
  - To locate the possible starting point



How to find a common subsequence

- Step 3 : Compare two IVSs frame by frame
  - Until a pair of frames are different



How to find a common subsequence

- We compare the frame by finding dissimilarity
  - Usually identical frames with dissimilarity <100
  - Two different frames with ~1000-2000
  - Computation time : ~10s

Dissimilarity

178

393

290

2430

We optimized the algorithm

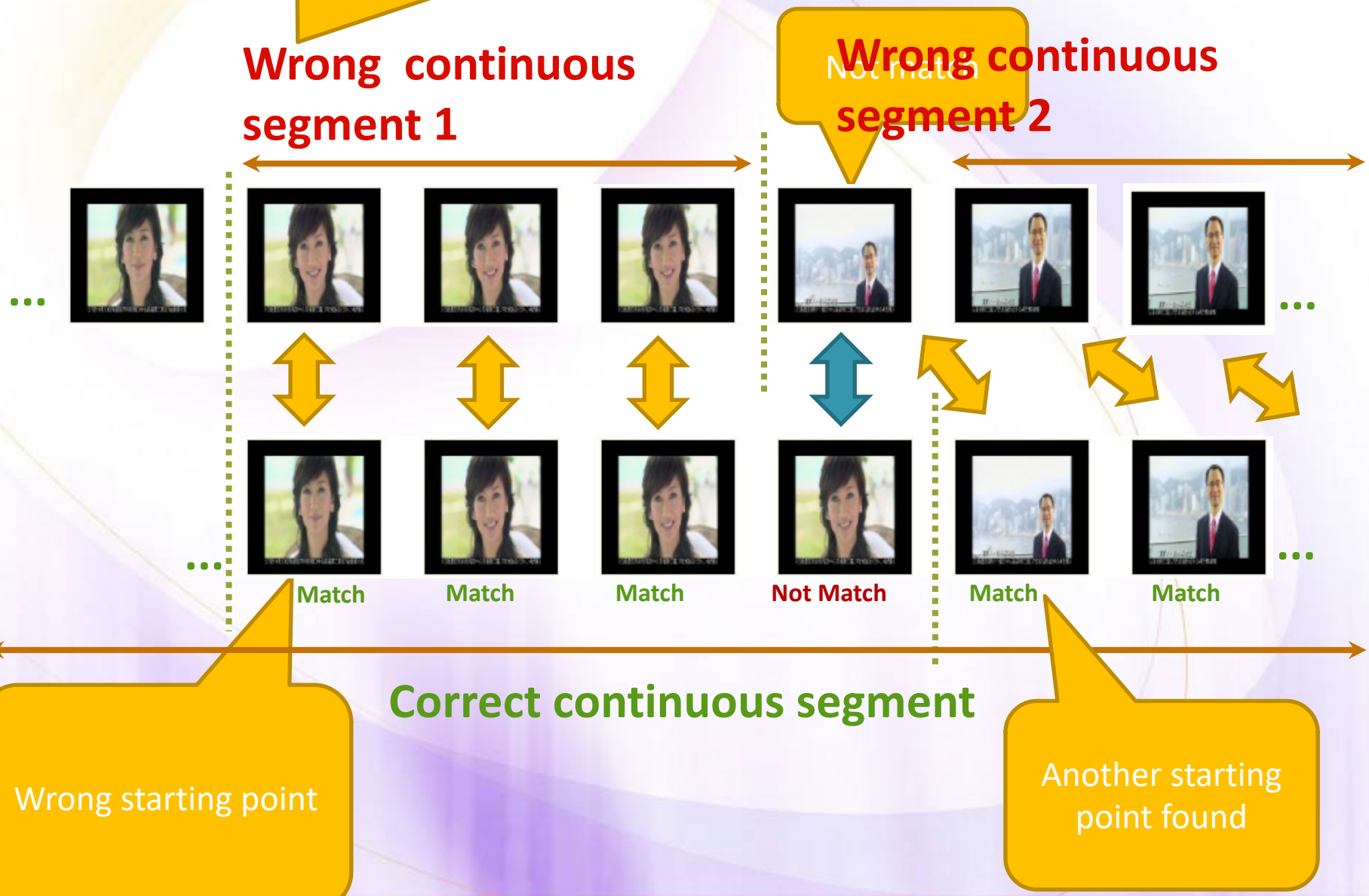
- The algorithm is provided by VIEW lab
- We put great effort on optimization of this algorithm.
  - More accuracy
  - Much faster

Fade in / fade out problem

- Fade in / out prevent us from finding a correct starting point.
- Usually has 1-2 frame shift

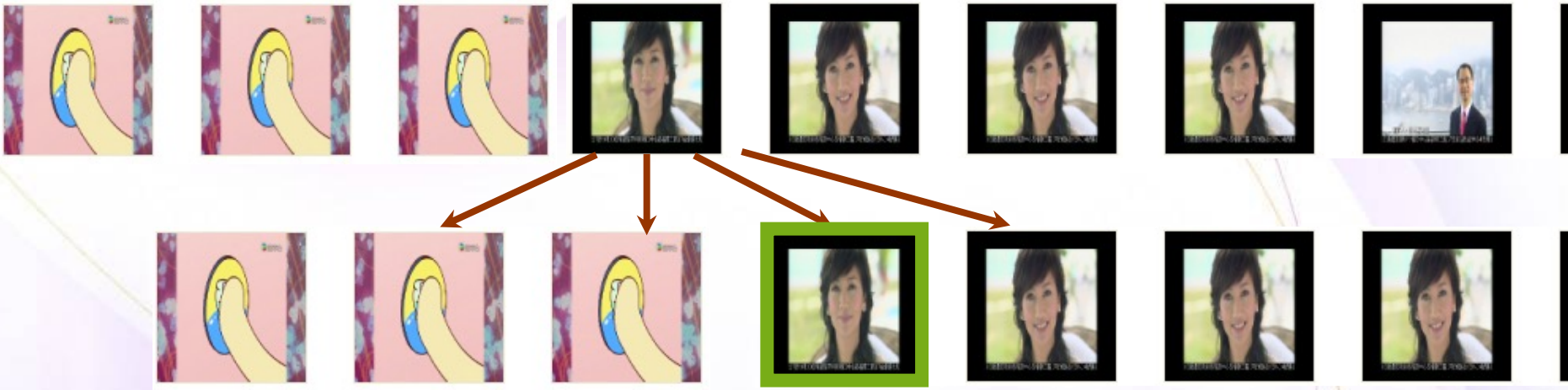
What

Larger dissimilarity with frame shift



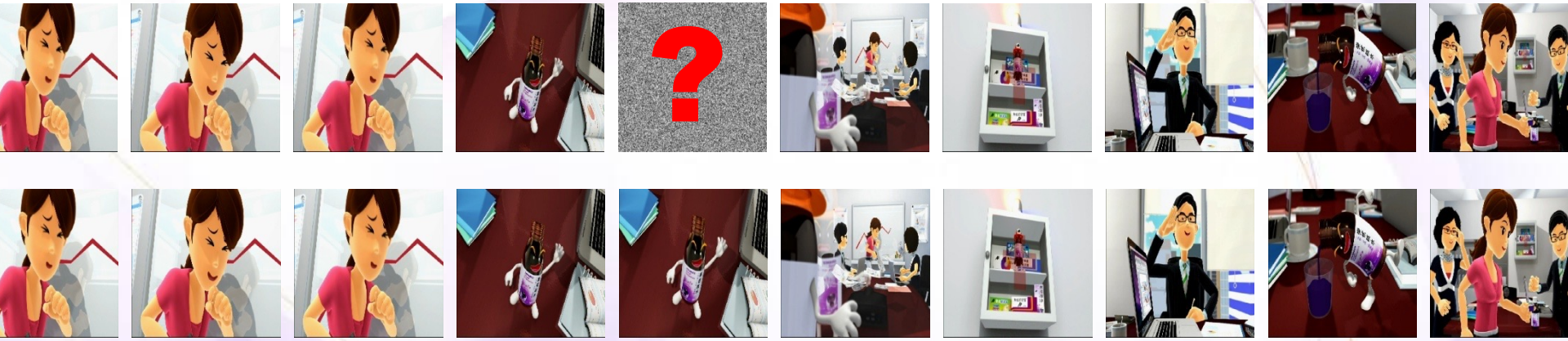
Dynamic range comparison

- Compare more than one frame until a acceptable frame found



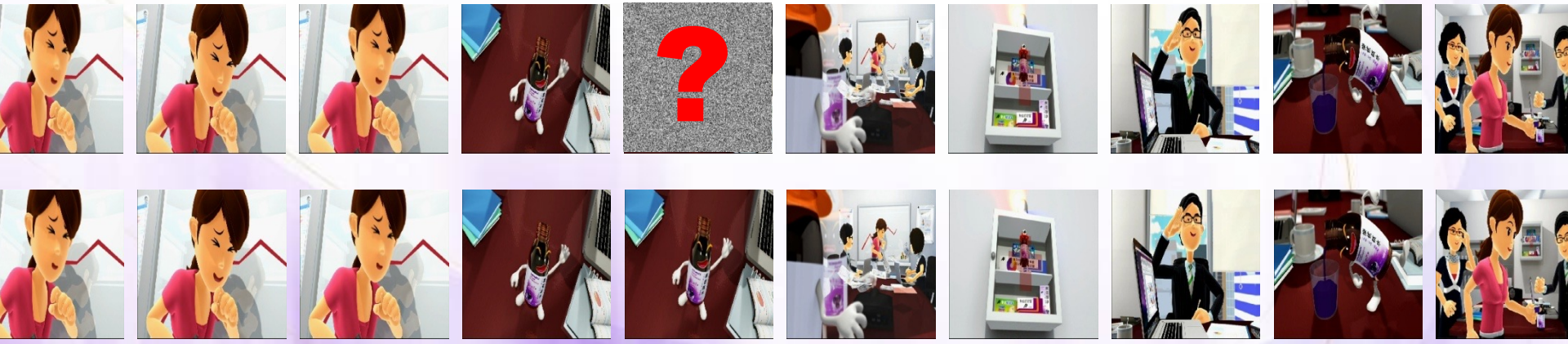
Noise frames

- Old algorithm can not handle the case if some frames crashed or with some noise



Dissimilarity of groups

- Instead of finding the common segments only, we group all possible segments together.
- Advertisement can be identify even with noise
- Preprocessing is required



Sample output

Improved algorithm

```
 1)   1 segments   dissimilarity :   112 len:254
 2)   1 segments   dissimilarity :    86 len:240
 3)   3 segments   dissimilarity :  2049 len:166 none-continuous
 4)   1 segments   dissimilarity :    36 len:122
 5)   1 segments   dissimilarity :    92 len:320
 6)   1 segments   dissimilarity :   112 len:250
 7)   1 segments   dissimilarity :    55 len:122
 8)   1 segments   dissimilarity :    68 len:363
 9)   1 segments   dissimilarity :   172 len:677
10)   1 segments   dissimilarity :   481 len:118
11)   1 segments   dissimilarity :   480 len:118
12)   1 segments   dissimilarity :    53 len:493
13)   1 segments   dissimilarity :    68 len:363
14)   1 segments   dissimilarity :    69 len:733
15)   1 segments   dissimilarity :    80 len:260
16)   1 segments   dissimilarity :   272 len: 44
17)   1 segments   dissimilarity :    27 len:124
18)   1 segments   dissimilarity :    70 len:369
19)   2 segments   dissimilarity :   376 len: 93 none-continuous
20)   3 segments   dissimilarity :   264 len:986 none-continuous
21)   1 segments   dissimilarity :    69 len:492
22)   1 segments   dissimilarity :    74 len:493
23)   1 segments   dissimilarity :   132 len:697
24)   1 segments   dissimilarity :    86 len:240

Saving to SQL server
----------------------------------------

Debugger stopped.
Program exited with status value:0.
```

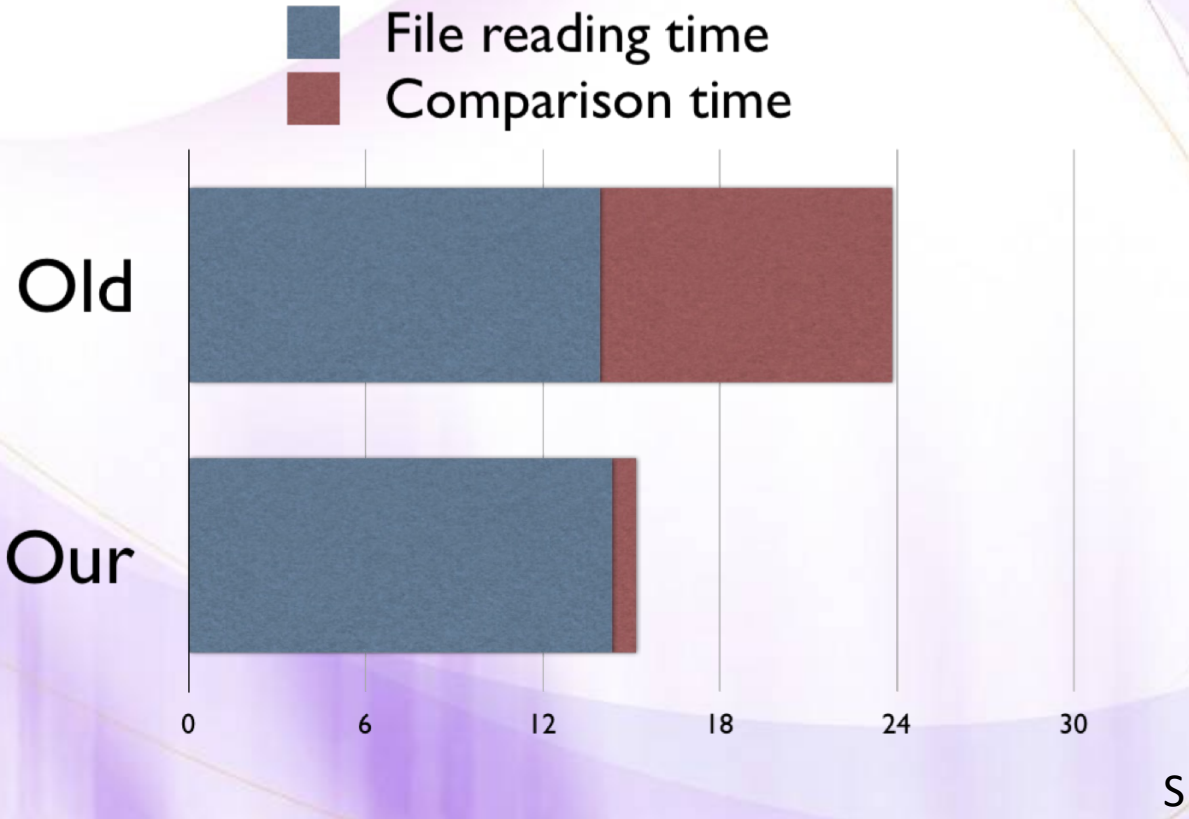
Previous algorithm

```
 1)   1 segments   dissimilarity :   112 len:254
 2)   1 segments   dissimilarity :    86 len:240
 3)   3 segments   dissimilarity :  2049 len:166
 4)   1 segments   dissimilarity :    36 len:122
 5)   1 segments   dissimilarity :    92 len:320
 6)   2 segments   dissimilarity :   112 len:247
 7)   1 segments   dissimilarity :    55 len:122
 8)   1 segments   dissimilarity :    68 len:363
 9)   2 segments   dissimilarity :   172 len:675
10)   1 segments   dissimilarity :   481 len:118
11)   1 segments   dissimilarity :   480 len:118
12)   1 segments   dissimilarity :    53 len:493
13)   1 segments   dissimilarity :    68 len:363
14)   1 segments   dissimilarity :    69 len:733
15)   1 segments   dissimilarity :    80 len:260
16)   1 segments   dissimilarity :   272 len: 44
17)   1 segments   dissimilarity :    27 len:124
18)   1 segments   dissimilarity :    70 len:369
19)   2 segments   dissimilarity :   376 len: 93
20)   3 segments   dissimilarity :   264 len:986
21)   1 segments   dissimilarity :    69 len:492
22)   1 segments   dissimilarity :    74 len:493
23)   1 segments   dissimilarity :   132 len:697
24)   1 segments   dissimilarity :    86 len:240

Saving to SQL server
----------------------------------------

Debugger stopped.
Program exited with status value:0.
```

False negative

Computation time improvement

- Some other improvement has been done
- Computation time is ~10 time faster



How we make it 10 times faster?

- In stead of directly find the common length, we further group the common segment into to larger groups to greatly reduce the size of n.
- We handle the groups one by one.
  - Divide n to many small n.
- Some coding style improvement.

Conclusion

- We learnt the cloud computing technologies
  - Cloud Computing
  - MapReduce & HDFS
- We improved & solved the scale up problem
  - By what we learnt
  - IVS algorithm
- We developed a MapReduce work flow
  - Provided huge flexibilities
  - Could be applied to general applications
  - 7*24 working

Why Hadoop

- Fault tolerant
  - Self healing
  - Data replicates
- Scalability
  - Processing power
  - Storage

Further development

- Improve the I/O performance
- Improve the MapReduce application
- Simplify the work flow
- Enhance the Web application
- By changing the core component, we have another similar application.
  - Surveillance video searching

Q & A

Limitation

- Poor CPU of the processing unit, makes the system complex.
- Hadoop cluster machine(Pentium 4 with 1GB RAM and 60GB HHD)
  - With 2GB RAM, the system can be two time faster
  - With 100MBytes/s switch, 3 time faster.
  - With dual core CPU, run two job at the same time and eliminate the overhead time of Hadoop.