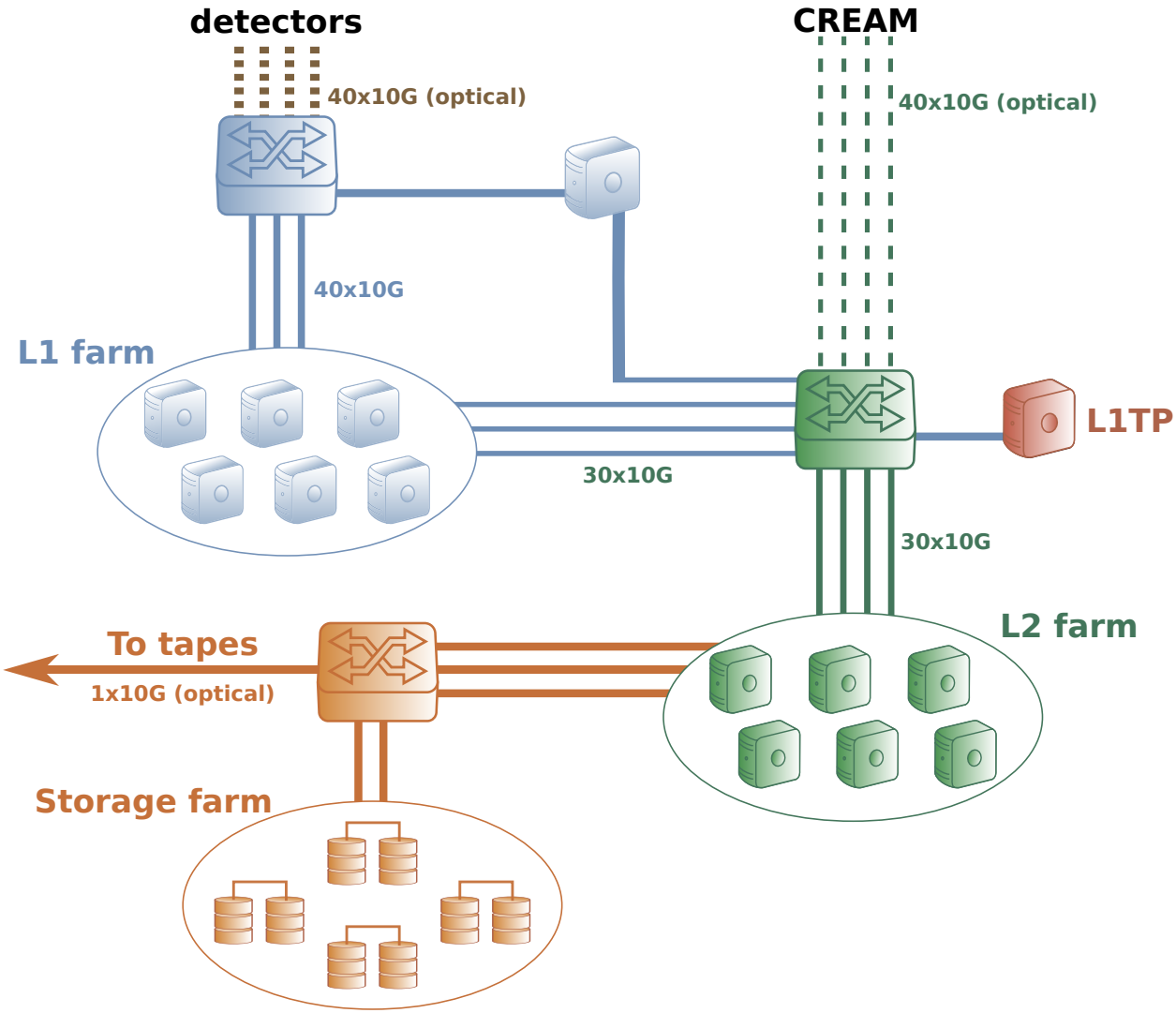
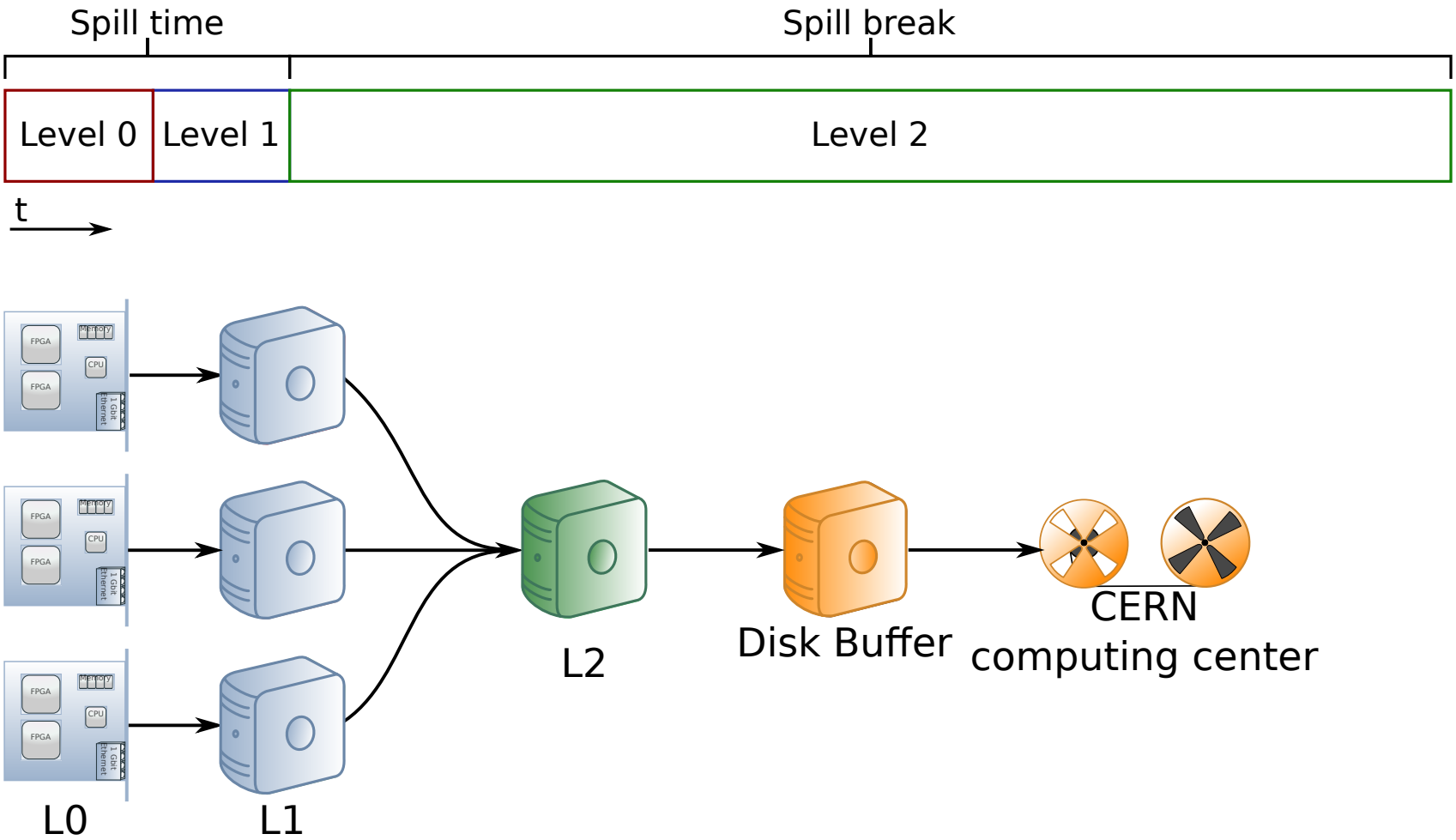
Infrastructure planned so far



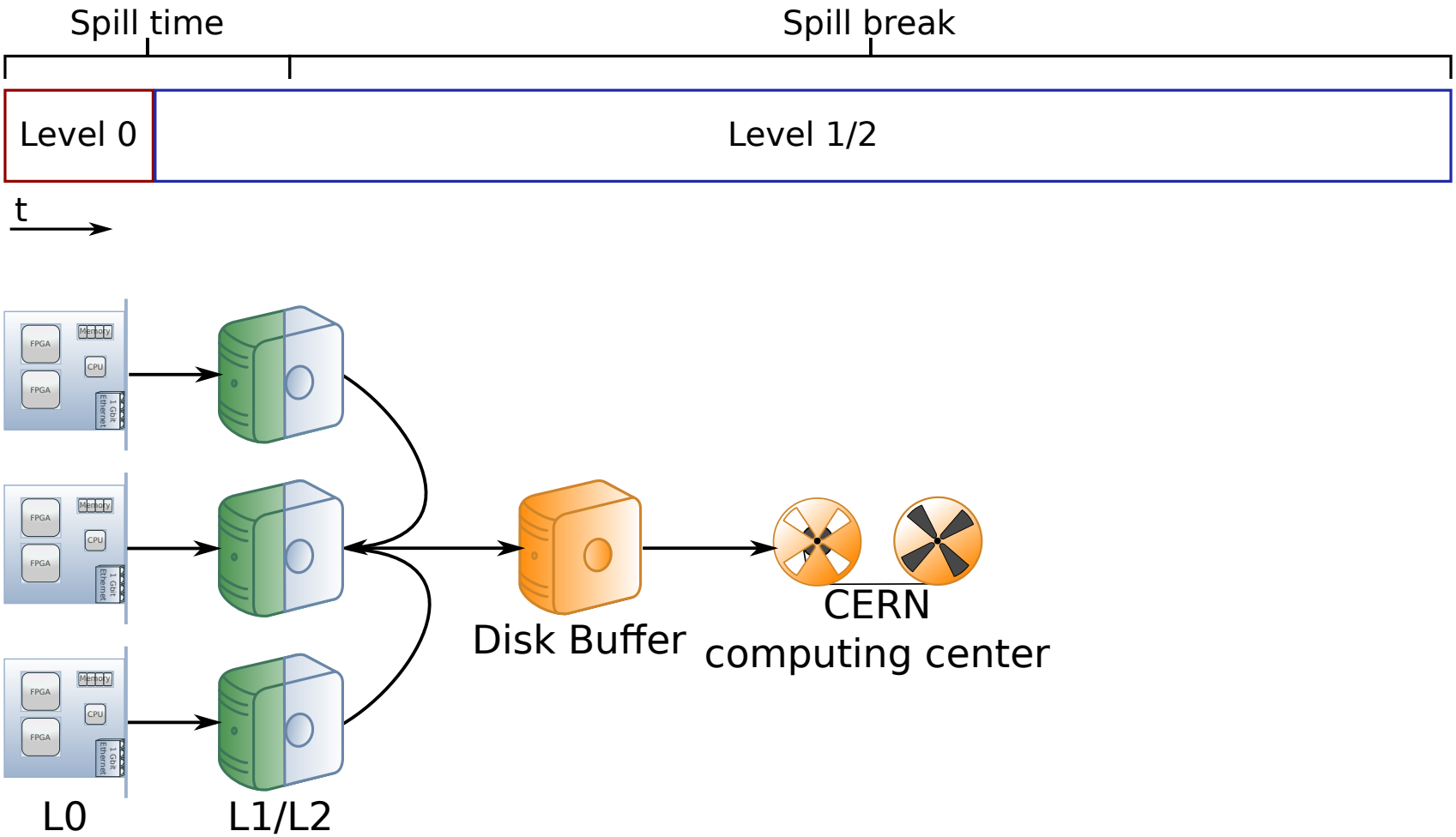
Logical data flow so far

Waste of resources?!

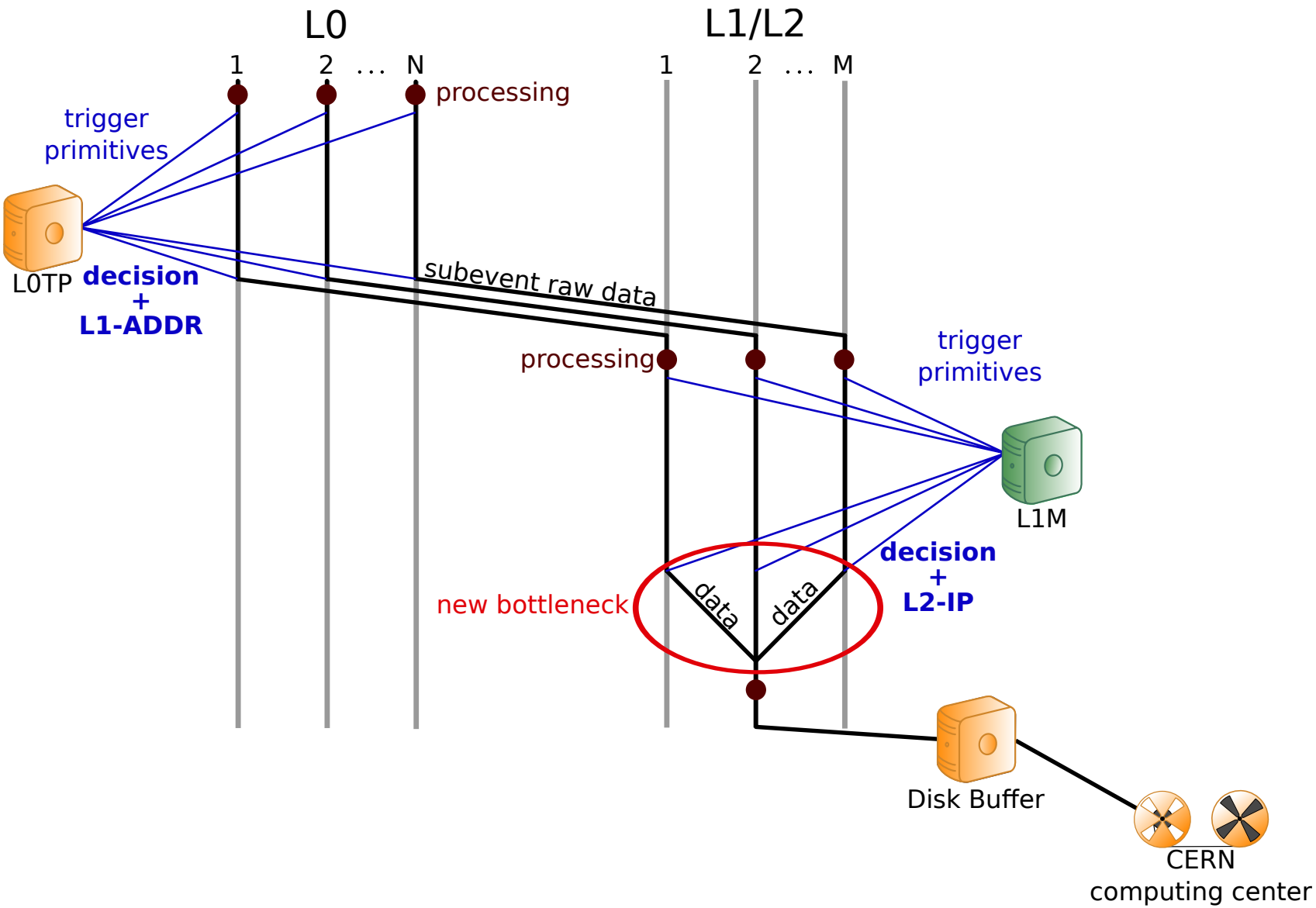


Merge L1 and L2

More efficient load distribution



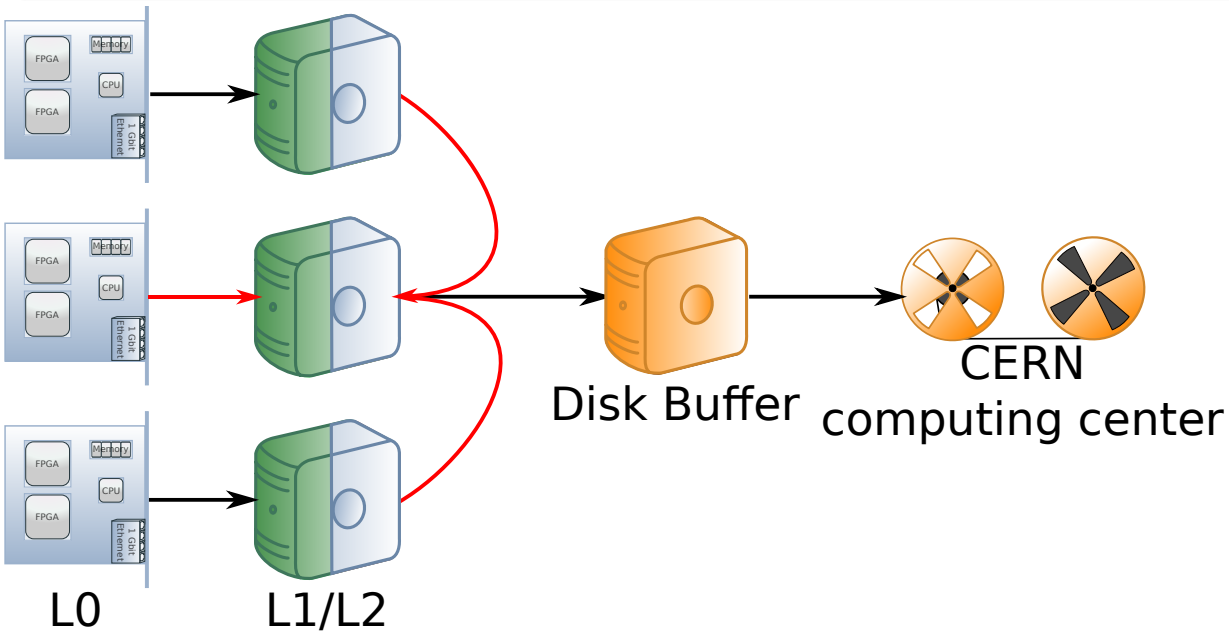
Data flow and trigger decision



Problems merging L1 and L2

Now you have two incoming data flows for one Cluster

Each PC now has to process L1 and L2 and you need a good load balancing so that L1 still can process the L0 data quick enough.



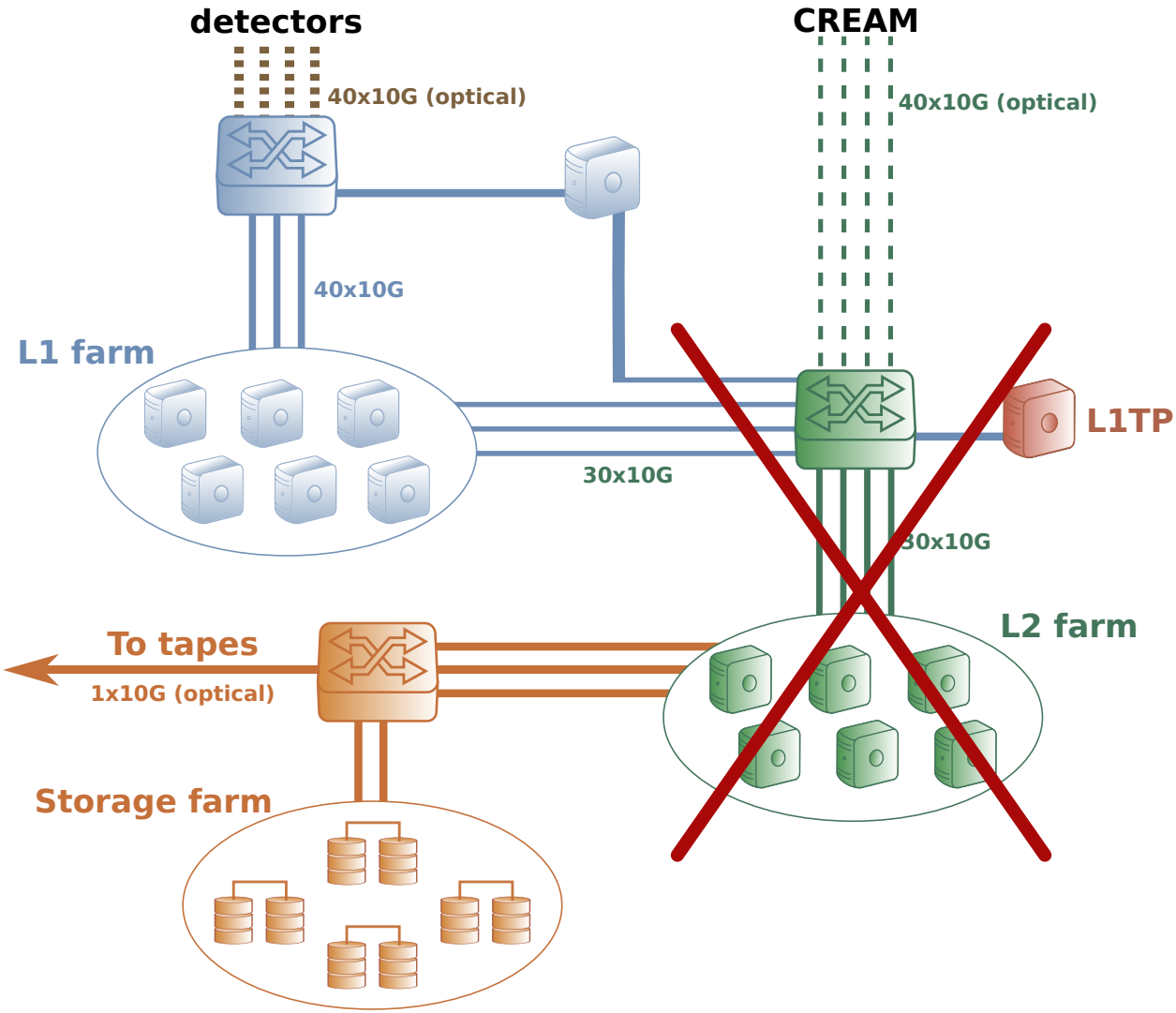
Caching solves bottleneck and disposes additional switch

To reduce network traffic: Don't start EB/L2 before L1 finished

- Need to store $\approx 100kHz * 5kB = 500MBps$ data in memory during spill (see table 61 in TDR) ! That's negligible.

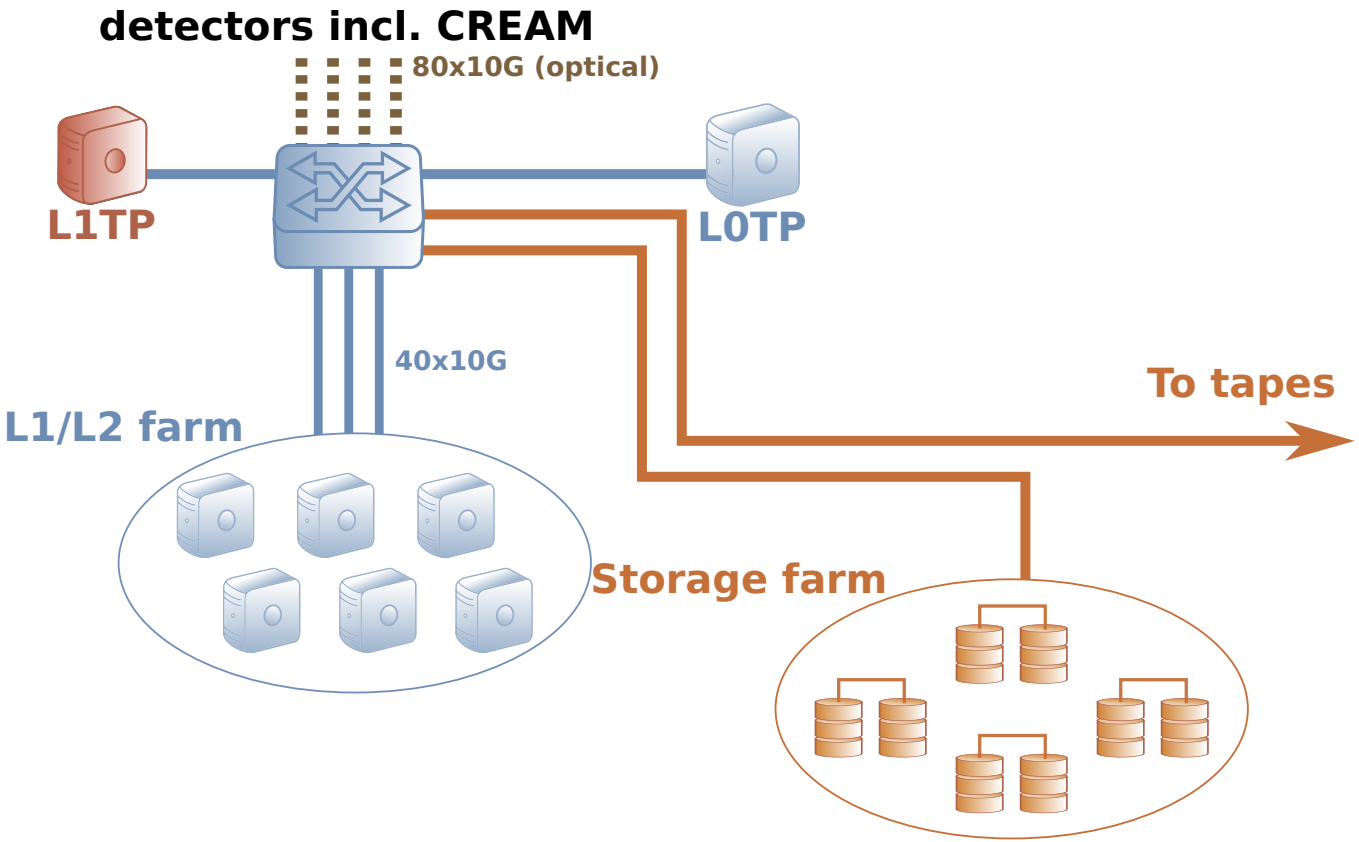
By holding data in cache until L1 has finished you need only one switch as you can use it for L0-L1 and L1-L2 transmission at separate times. Than the merged cluster also doesn't need to be bigger than the old L1 or L2 cluster ($\approx$ 50% less PCs)

Overview with L1 and L2 separated



Overview with L1 and L2 merged

Need one huge core switch



Proposals by www.SEiCOM-muc.de

Arista 7504

- Chassis + Supervisor: 44,400€
- 3*48-port linecards: 67,638€
- Redundant supervisor: 6,829€
- 60*SFP+ SR (up to 220m): 21,780€

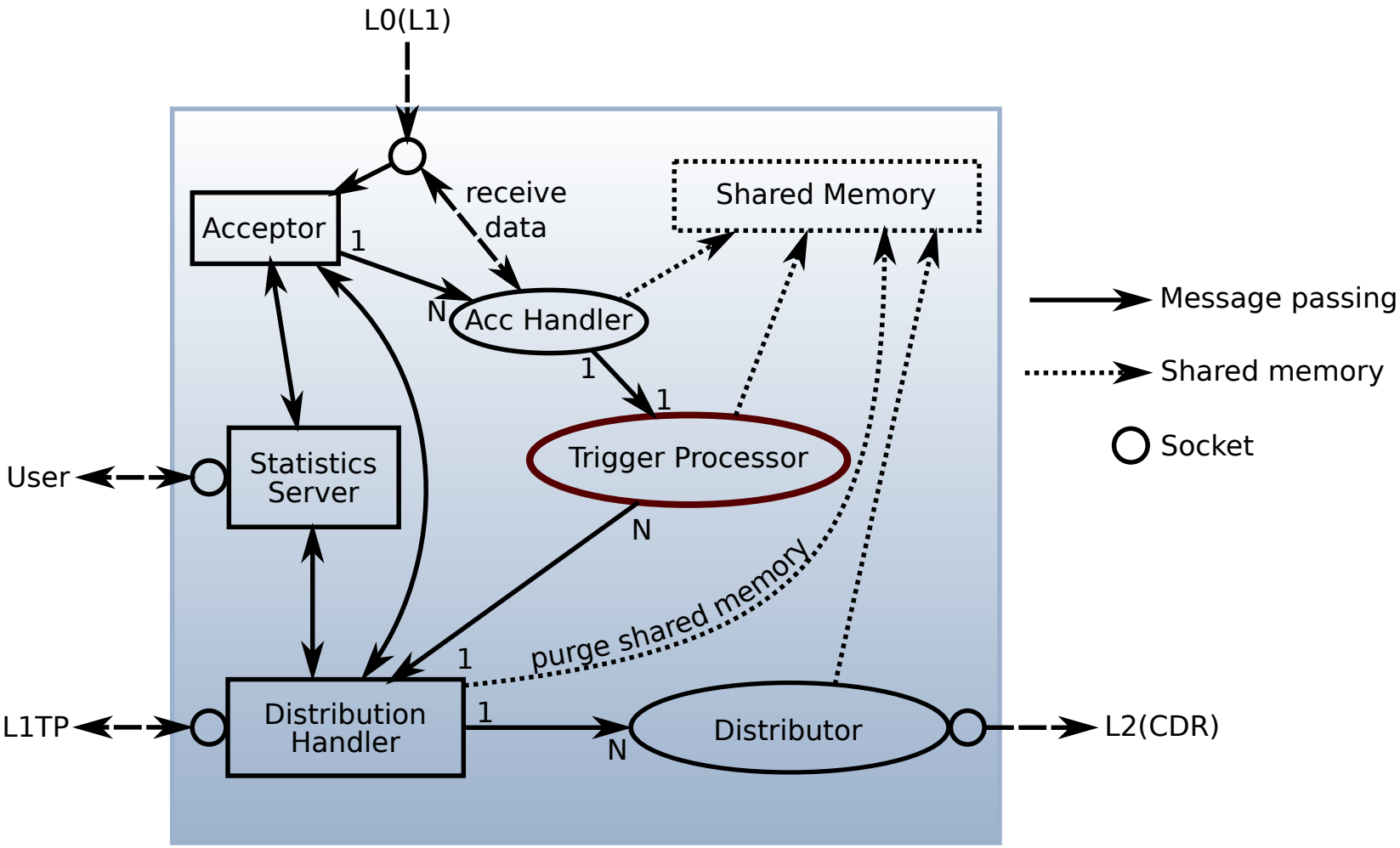
141 k€

Arista 7148S

- 3 *48 port switch: 44,400€
- 3*48-port Linecards: 67,638€
- 60*SFP+ SR (up to 220m): 21,780€
- Redundant power supply included

66 k€

Online farm framework



TriggerProcessor Class

Parent Class designed by me, inherited and implemented by the subdetector groups

```
TriggerPrimitive processData(char* data) const;
```

What will TriggerPrimitive Class look like?

```
std::size_t getEventID() const;
bool disposeEvent() const;
char* getReconstructedData() const;
```