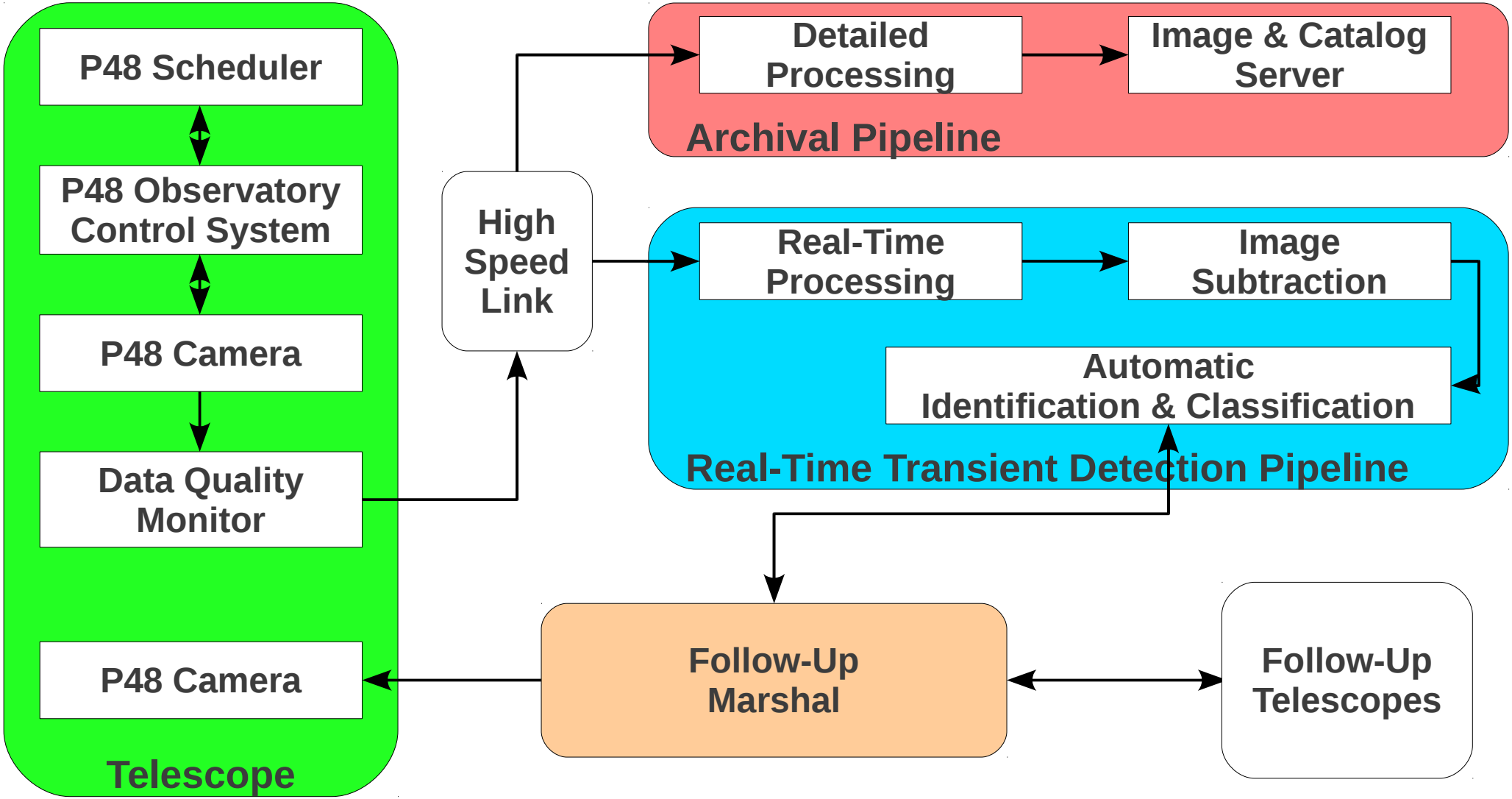
Implementing the Palomar Transient Factory Real-Time Detection Pipeline in GLADE: Results and Observations

Florin Rusu
University of California at Merced

Peter Nugent
Kesheng Wu
Lawrence Berkeley National Lab

Palomar Transient Factory (PTF)



- 2,000-4,000 images 2,048x4,096 pixels per night
- 60-100 GB per night
- Complete processing of an image set within 30-45 minutes of aquisition

Automatic Identification & Classification

- ~1-1.5 million candidates extracted per night; ~10,000 every 45 minutes; 30-150 are real
- Real-or-bogus classification
  - Is the candidate real?
- What is the transient type of a real candidate?
  - VarStar; SN/Nova; circumnuclear event; asteroid
- Machine learning classifiers
  - Probabilistic random forest + domain knowledge
  - Training with human experts
  - Human verification & confirmation

Real-Or-Bogus Classification

- Is the candidate real?
  - Realbogus score
- Random forest classifier
  - 28 features extracted from subtraction images
  - 5 classes: {bogus, suspect, unclear, maybe, realish}
  - Training dataset: 574 candidates manually labeled with realbogus score by human scanners
  - Realbogus score = weighted average of class probabilities
- Domain knowledge
  - Candidate appears in two subtraction images within 6 days
  - Contextual realbogus score = func (realbogus score, realbogus score(subtraction neighbors))

Real-Or-Bogus Processing

1) Identify candidates

```
SELECT s1.ujd, c1.id, c1.ra,
    c1.dec, c1.sub_id,
    c1.xint_new, c1.yint_new,
    c1.pos_sub
FROM subtraction s1 JOIN
    candidate c1 ON
    (c1.sub_id = s1.id) JOIN
    rb_classifier rbc ON
    (rbc.sub_id = c1.sub_id AND
     rbc.candidate_id = c1.id)
WHERE
     (s1.ujd > 2455844) AND
     (s1.ujd < 2455845) AND
     (rbc.realbogus > 0.17) AND
     (rbc.bogus < 0.35) AND
     (c1.b_image > 0.7) AND
     (c1.a_image < 3.0 OR
      c1.mag < 15.0) AND
     (c1.pos_sub = 'T')
```

Ad-hoc queries

2) Prune single appearance

```
SELECT COUNT(c1.id)
FROM subtraction s1 JOIN candidate c1 ON
    (c1.sub_id = s1.id) JOIN rb_classifier rbc
    ON (rbc.candidate_id = c1.id)
WHERE
    (c1.ra BETWEEN %1f and %2f) AND
    (c1.dec BETWEEN %3f and %4f) AND
    ((rbc.realbogus > %6f) OR
     (c1.pos_sub != 'T')) AND
    (c1.b_image > 0.7) AND
    (c1.a_image < 3.0 OR c1.mag < 15.0) AND
    ((s1.ujd BETWEEN %7f AND %8f) OR
     (s1.ujd BETWEEN %9f AND %10f))
```

Multiple related queries

3) Contextual realbogus

```
SELECT c.id, c.sub_id, c.mag,
    c.mag_ref, c.sym,
    c.xint_new, c.yint_new,
    c.a_image, c.b_image,
    ((c.xint_new - %1f)^2 +
     (c.yint_new - %2f)^2) ^0.5 AS
    dist, s.good_pix_area,
    rbc.realbogus
FROM subtraction s JOIN
    candidate c ON
    (c1.sub_id = s1.id) JOIN
    rb_classifier rbc ON
    (rbc.sub_id = c.sub_id AND
     rbc.candidate_id = c1.id)
WHERE
     c.sub_id = %3f AND
     c.pos_sub = 'T'
ORDER BY
     dist limit 20
+
Application code
```

Complex processing beyond SQL

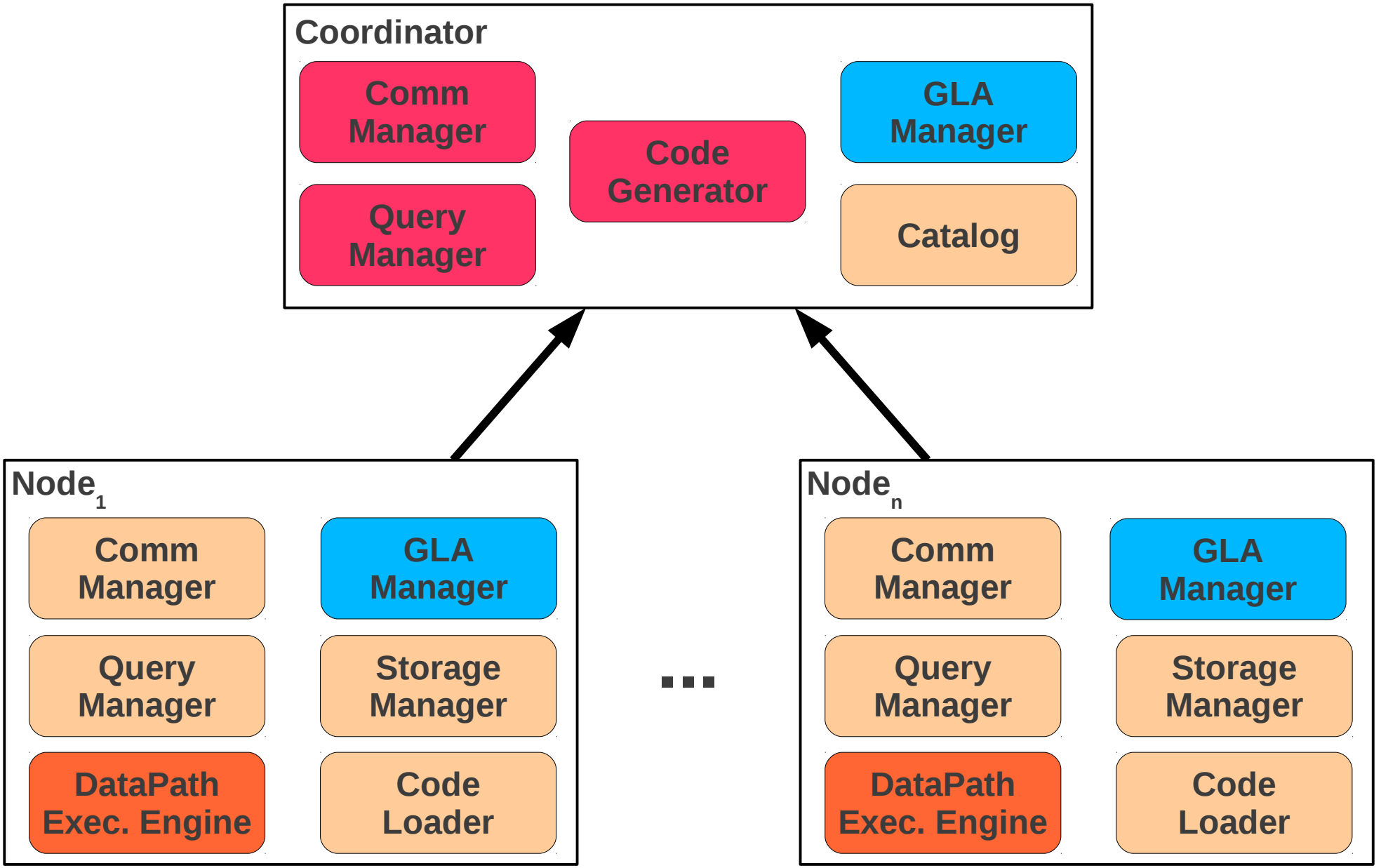
Table	# tuples	# attributes
subtraction	1,039,758	51
candidate	672,912,156	46
rb_classifier	672,906,737	9

What Is the Problem?

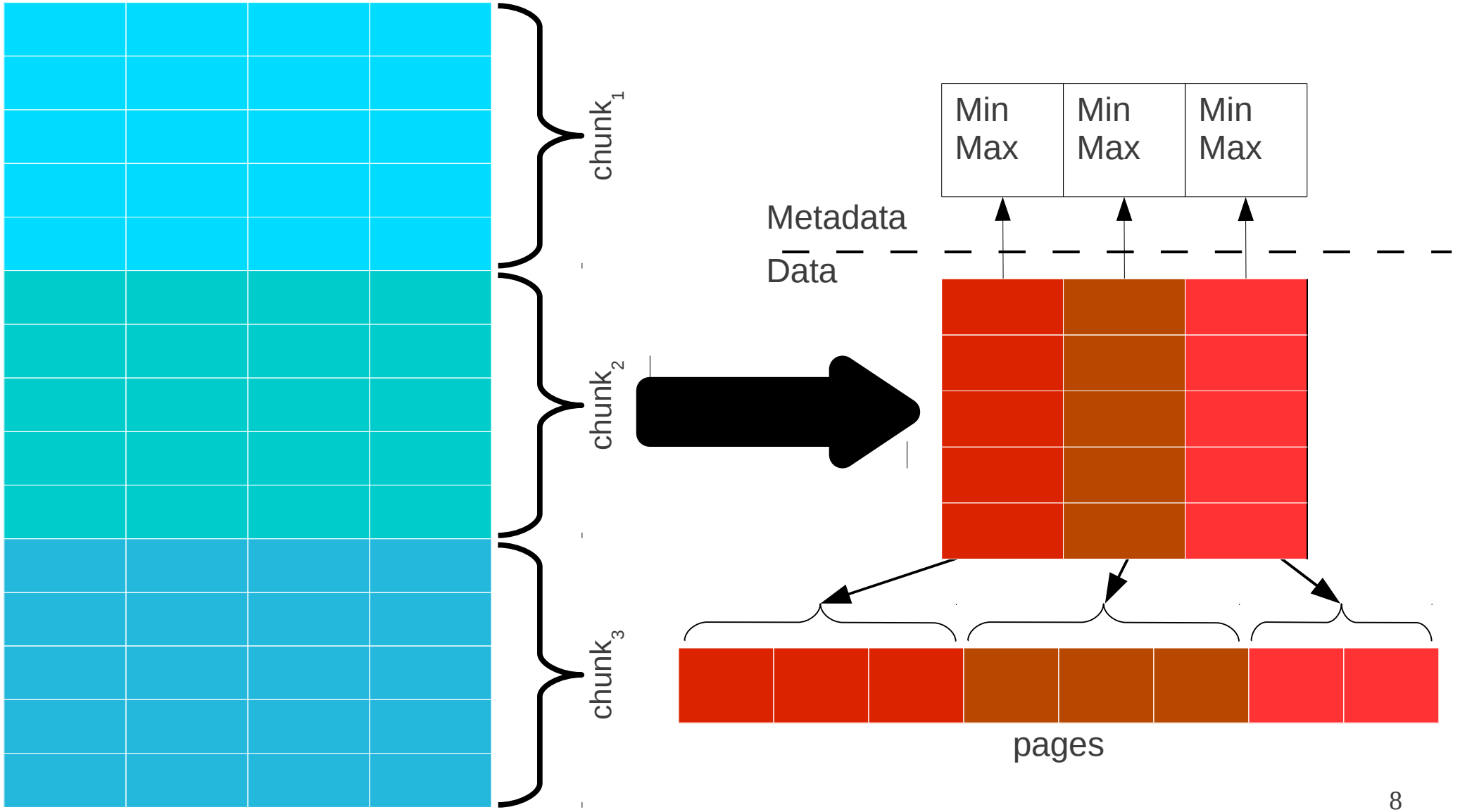
- PostgreSQL 8.4 + Python
- Raw data: 160 GB
- Indexes: subtraction (5); candidate (9); rb_classifier (2)
- October 10, 2011: 2,997 images; 1,939,059 candidates (647 per image); 40,087 identified; 5,600 non-singleton

Phase	PostgreSQL	PostgreSQL + indexes
Data ingestion	59 sec	**1 hour 8 sec**
Identification	45 sec	4.67 sec
Pruning	**607 hours**	15 min 39 sec
Contextual realbogus	68 hours	0.79 sec
Total	**675 hours**	**1 hour 16 min**

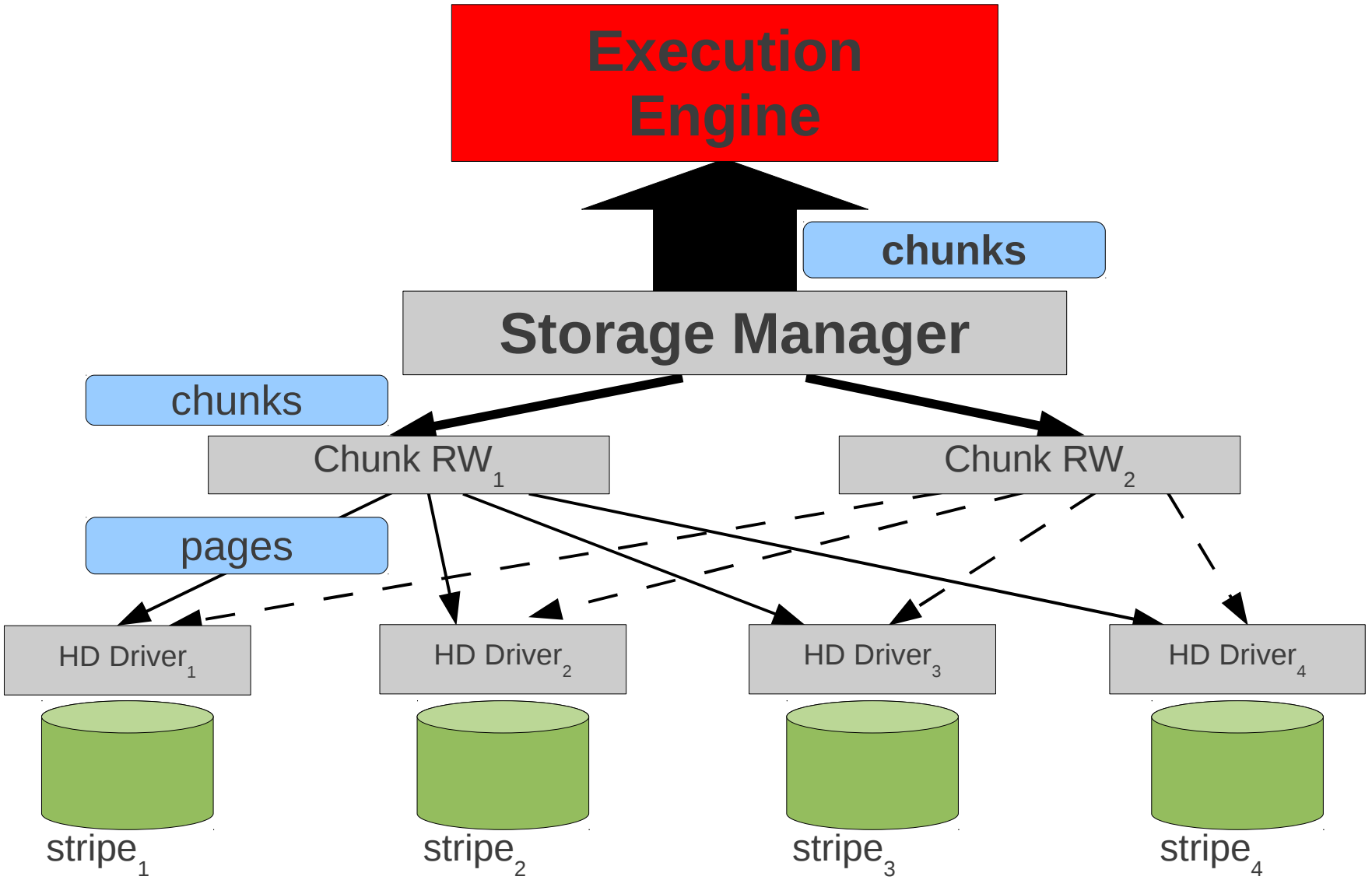
GLADE


Coordinator
Comm Manager
Query Manager
Code Generator
GLA Manager
Catalog
Node1
Comm Manager
Query Manager
DataPath Exec. Engine
GLA Manager
Storage Manager
Code Loader
...
Noden
Comm Manager
Query Manager
DataPath Exec. Engine
GLA Manager
Storage Manager
Code Loader

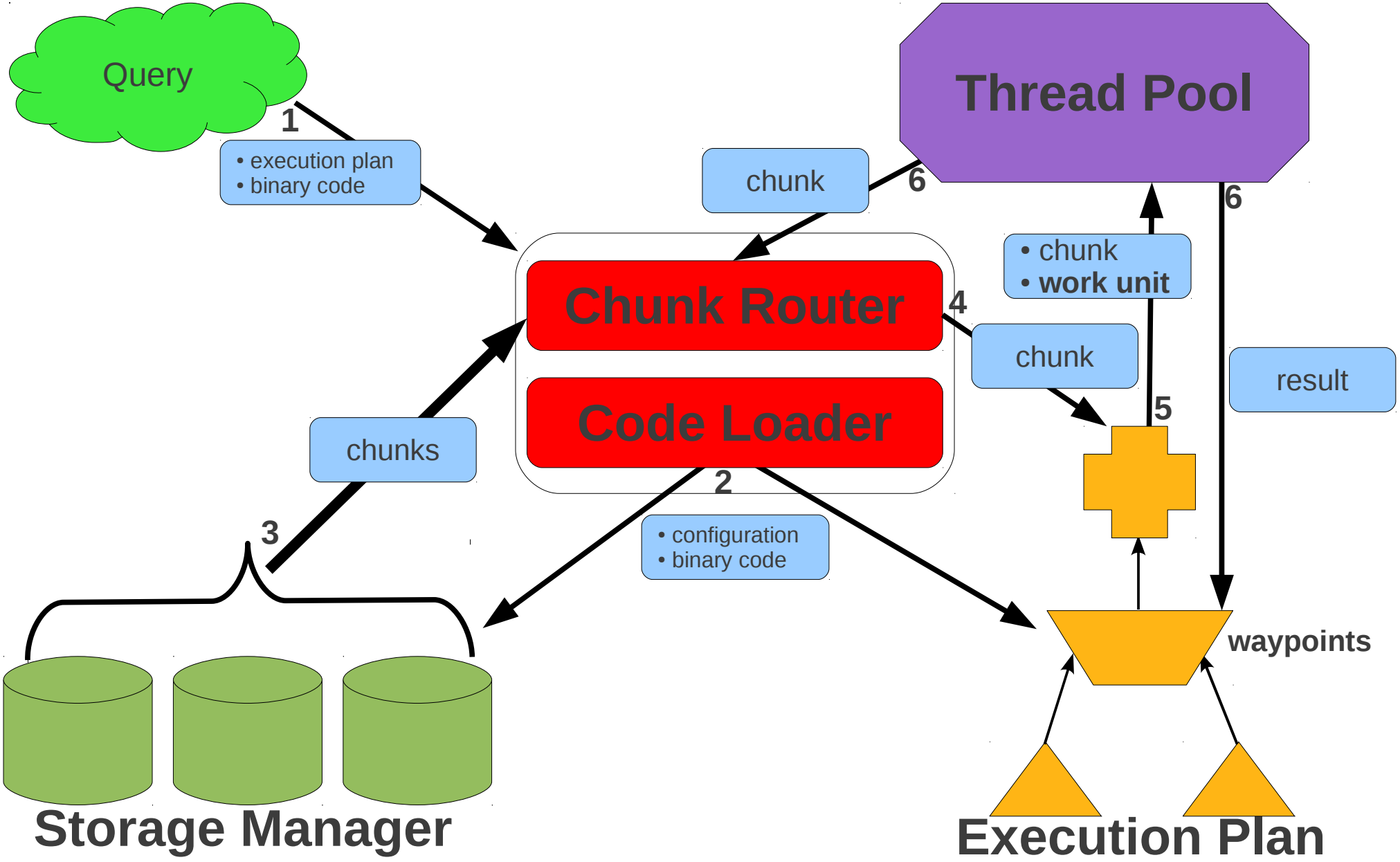
Data Organization



Storage Manager

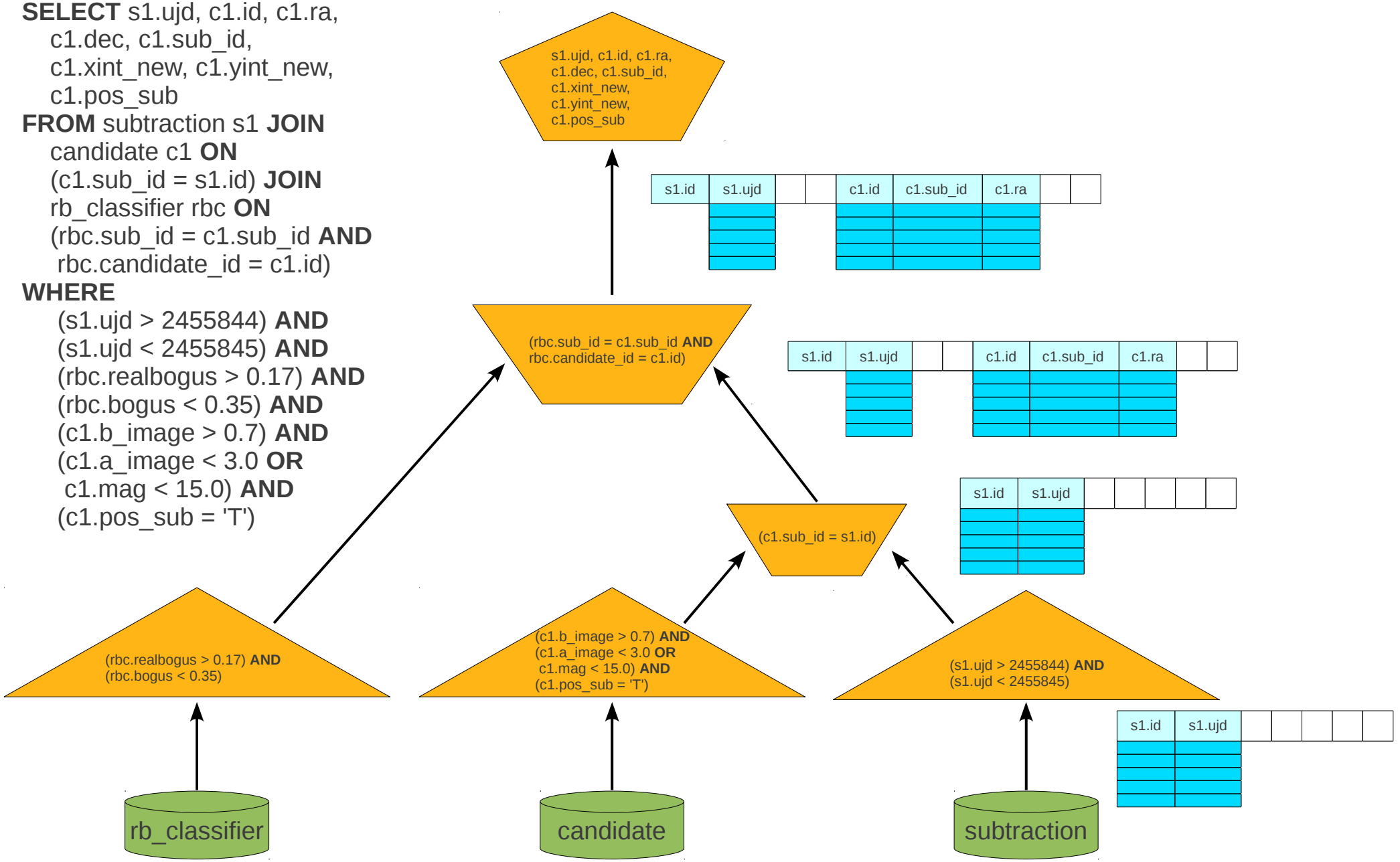


Execution Engine



Query Processing

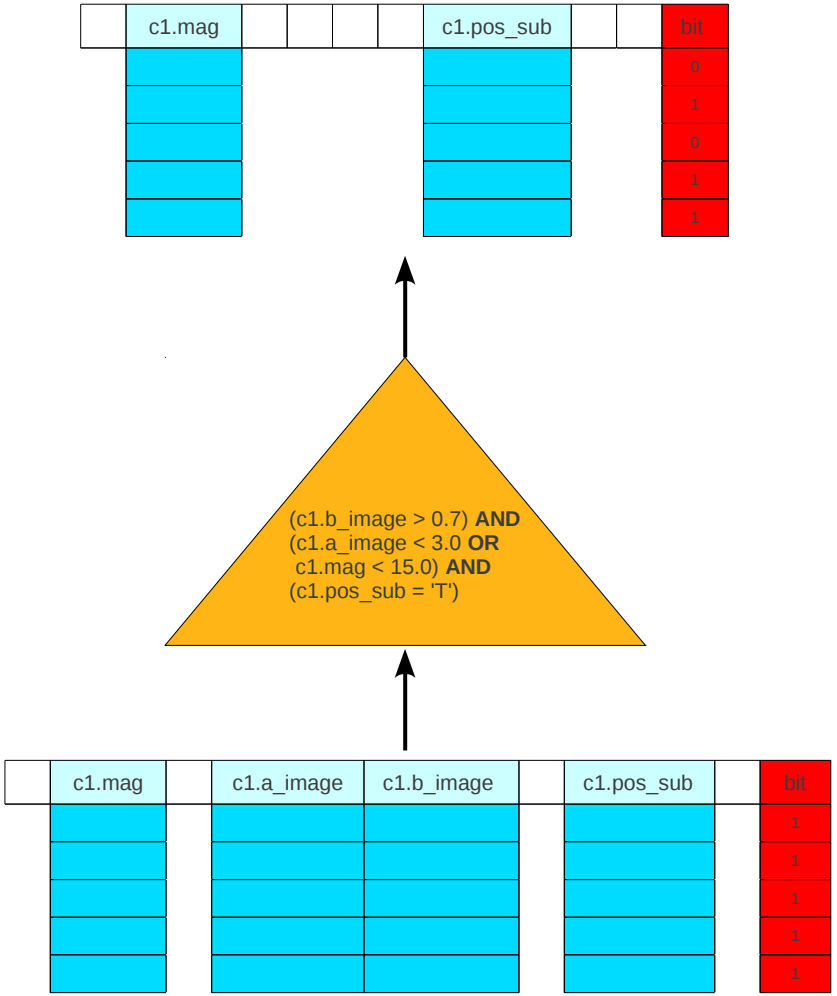
```
SELECT s1.ujd, c1.id, c1.ra,
    c1.dec, c1.sub_id,
    c1.xint_new, c1.yint_new,
    c1.pos_sub
FROM subtraction s1 JOIN
    candidate c1 ON
    (c1.sub_id = s1.id) JOIN
    rb_classifier rbc ON
    (rbc.sub_id = c1.sub_id AND
     rbc.candidate_id = c1.id)
WHERE
     (s1.ujd > 2455844) AND
     (s1.ujd < 2455845) AND
     (rbc.realbogus > 0.17) AND
     (rbc.bogus < 0.35) AND
     (c1.b_image > 0.7) AND
     (c1.a_image < 3.0 OR
      c1.mag < 15.0) AND
     (c1.pos_sub = 'T')
```



Code Generator & Code Loader

- Generate query-specific C/C++ code for each node in query execution tree
  - Hard-code (no interpretation, optimal)
  - M4 template with C code embedded for each operator
    - SELECTION, JOIN, PRINT, DISK SCANNER
- Compile with C/C++ compiler
  - Generate dynamic library
- Dynamically load code at runtime
- Execute inside Execution Engine, near data

Selection



Waypoint

Stateless

Process work unit (chunk)

1. Extract columns from chunk
2. Memory-map columns to arrays of column type
3. For all tuples in chunk
4. If (selection cond is false)
5. Set bit to false
6. Drop columns not used anymore
7. Add columns back to chunk

M4 code

```
for (int i = 0; i < numTuples; i++) {
  // extract values of attributes from streams
  m4_set_foreach([ATTS_ALL], [_A_], [dnl
    // extracting _A_
    M4_ATT_TYPE(_A_) _A_;
    _A_= M4_ATT_DATA(_A_)[[i]];
  ]])dnl

  dnl # selection code for all the predicates
  If (!(M4_REM_VAL(m4_second(_P_))))
    bits.ResetBit(i);
}
```

C++ code

```
for (int i = 0; i < numTuples; i++) {
  // extract values of attributes from streams

  // extracting candidate_b_image
  FLOAT candidate_b_image = data_candidate_b_image[i];
  // extracting candidate_a_image
  FLOAT candidate_a_image = data_candidate_a_image[i];
  // extracting candidate_mag
  FLOAT candidate_mag = data_candidate_mag[i];
  // extracting candidate_pos_sub
  CHAR<1> candidate_pos_sub = data_candidate_pos_sub[i];

  if (!(((candidate_b_image > 0.7) &&
        (candidate_a_image < 3.0 || candidate_mag < 15.0) &&
        (candidate_pos_sub == "T"))))
    bits.ResetBit(i);
}
```

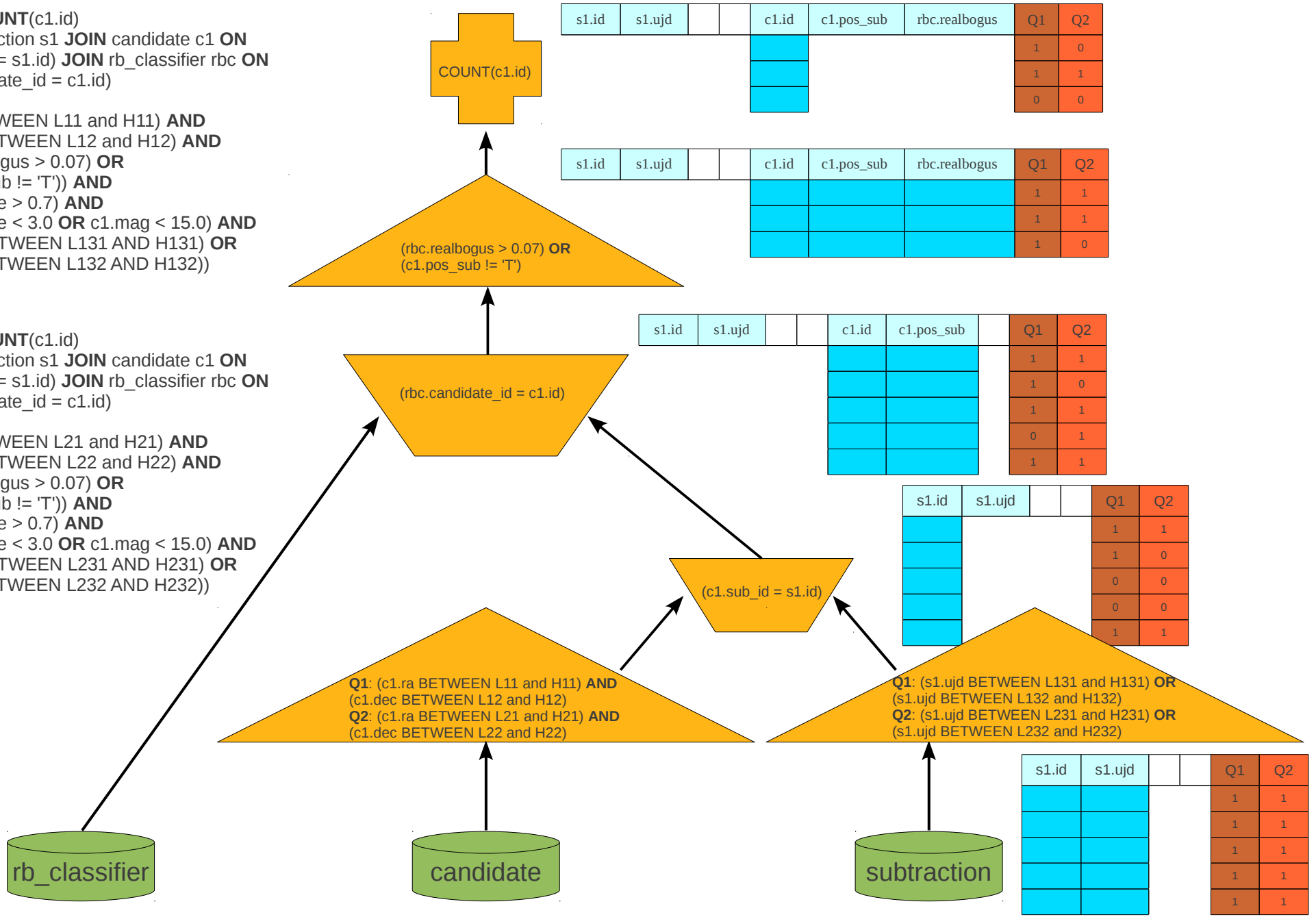
Multi-Query Processing

Q1

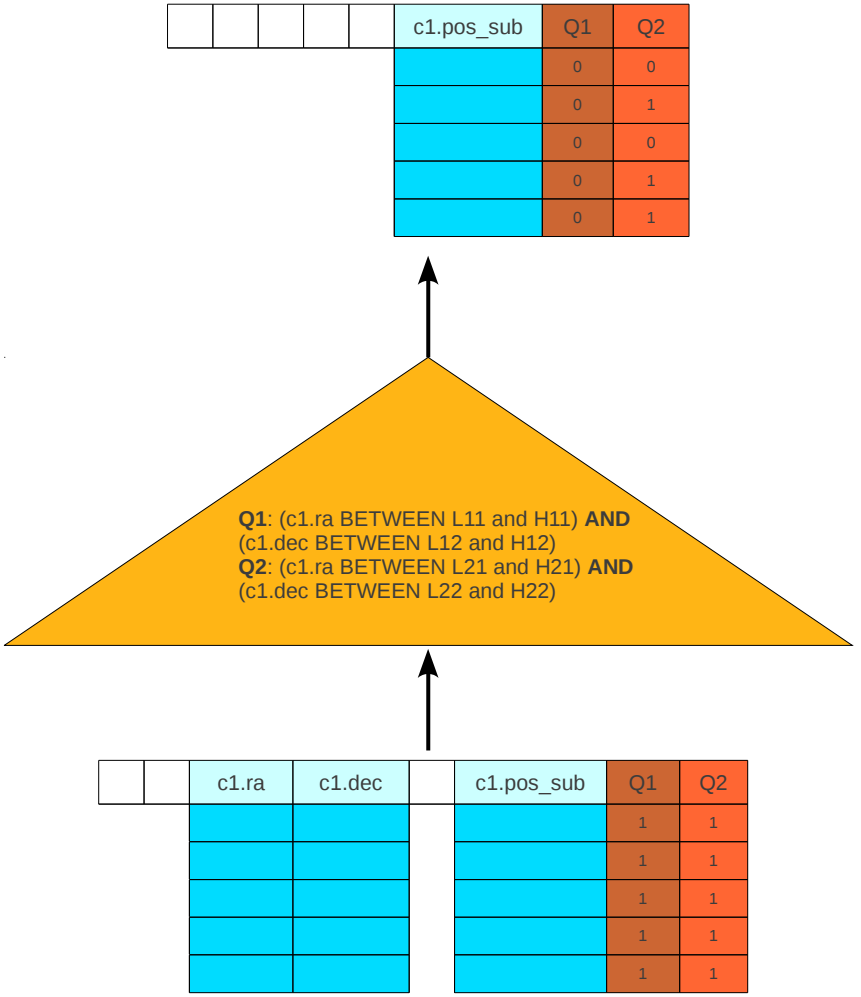
```
SELECT COUNT(c1.id)
FROM subtraction s1 JOIN candidate c1 ON
    (c1.sub_id = s1.id) JOIN rb_classifier rbc ON
    (rbc.candidate_id = c1.id)
WHERE
    (c1.ra BETWEEN L11 and H11) AND
    (c1.dec BETWEEN L12 and H12) AND
    ((rbc.realbogus > 0.07) OR
     (c1.pos_sub != 'T')) AND
    (c1.b_image > 0.7) AND
    (c1.a_image < 3.0 OR c1.mag < 15.0) AND
    ((s1.ujd BETWEEN L131 AND H131) OR
     (s1.ujd BETWEEN L132 AND H132))
```

Q2

```
SELECT COUNT(c1.id)
FROM subtraction s1 JOIN candidate c1 ON
    (c1.sub_id = s1.id) JOIN rb_classifier rbc ON
    (rbc.candidate_id = c1.id)
WHERE
    (c1.ra BETWEEN L21 and H21) AND
    (c1.dec BETWEEN L22 and H22) AND
    ((rbc.realbogus > 0.07) OR
     (c1.pos_sub != 'T')) AND
    (c1.b_image > 0.7) AND
    (c1.a_image < 3.0 OR c1.mag < 15.0) AND
    ((s1.ujd BETWEEN L231 AND H231) OR
     (s1.ujd BETWEEN L232 AND H232))
```



Multi-Query Selection



M4 code

```
for (int i = 0; i < numTuples; i++) {
  // extract queries for current tuple
  QueryIDSet qry=queries[[i]];
  qry.Intersect(queriesToRun);

  // extract values of attributes from streams
  m4_set_foreach([ATTS_ALL], [_A_], [dnl
    // extracting _A_
    M4_ATT_TYPE(_A_) _A_;
    if (_A_[]_Qrys.Overlaps(queriesToRun)) {
      _A_= M4_ATT_DATA(_A_)[[i]];
    }
  ]])dnl

  dnl # selection code for all the predicates
  m4_foreach([_P_], [M4_Predicates], [dnl
    // do M4_QUERY_NAME(_P_)
    if (qry.Overlaps(M4_QUERY_NAME(_P_))
        && !(M4_REM_VAL(m4_second(_P_))))
      qry.Difference(M4_QUERY_NAME(_P_));
  ])dnl

  queries[[i]] = qry;
}
```

C++ code

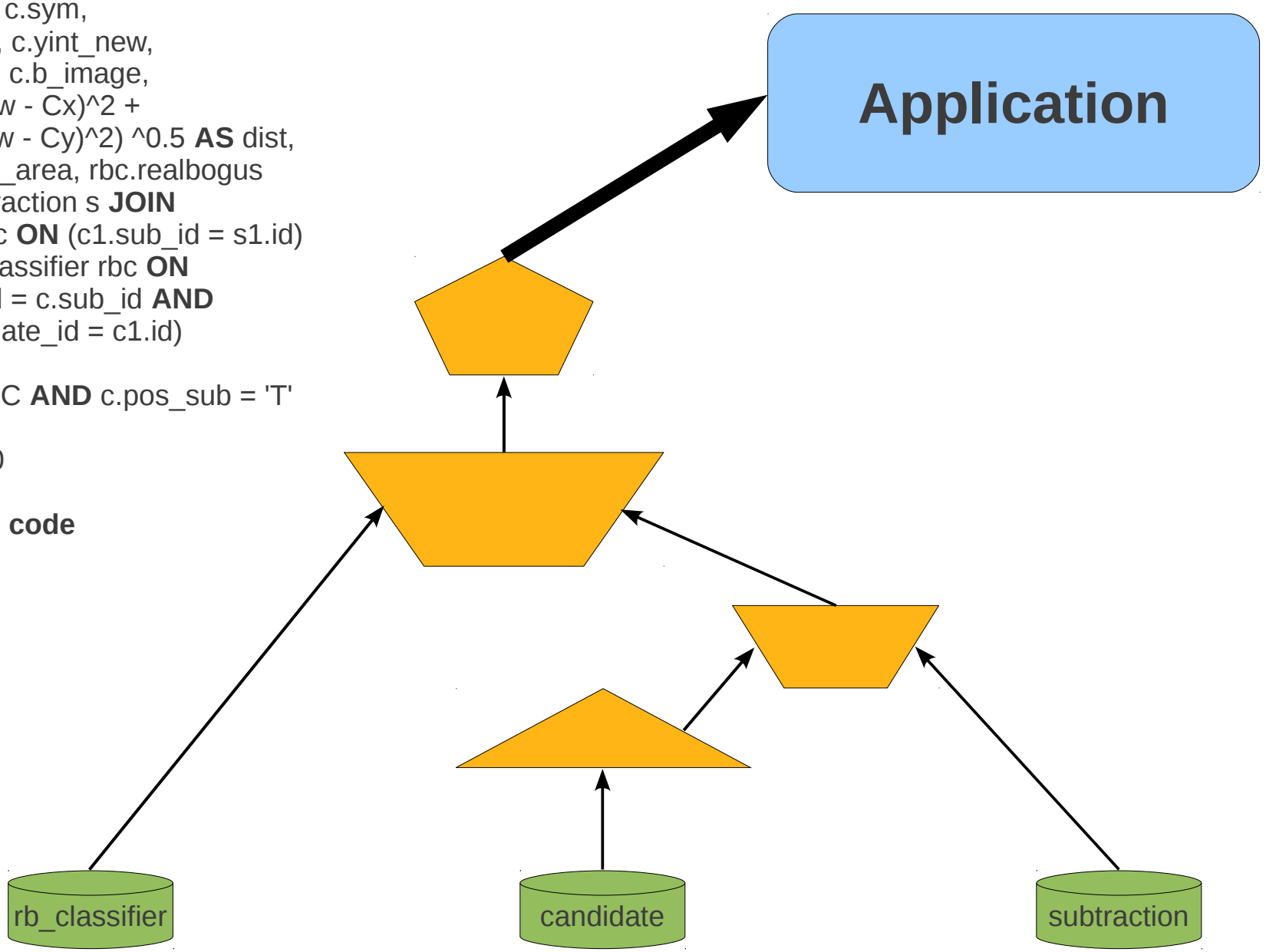
```
for (int i = 0; i < numTuples; i++) {
  // extract queries for current tuple
  QueryIDSet qry=queries[[i]];
  qry.Intersect(queriesToRun);

  // extract values of attributes from streams

  // extracting candidate_ra
  FLOAT candidate_ra;
  if (candidate_ra_Qrys.Overlaps(queriesToRun))
    candidate_ra = data_candidate_ra[i];
  // extracting candidate_dec
  FLOAT candidate_dec;
  if (candidate_dec_Qrys.Overlaps(queriesToRun))
    candidate_dec = data_candidate_dec[i];
  // extracting candidate_b_image
  FLOAT candidate_b_image;
  if (candidate_b_image_Qrys.Overlaps(queriesToRun))
     candidate_b_image = data_candidate_b_image[i];
  // extracting candidate_a_image
  FLOAT candidate_a_image;
  if (candidate_a_image_Qrys.Overlaps(queriesToRun))
    candidate_a_image = data_candidate_a_image[i];
  // extracting candidate_mag
  FLOAT candidate_mag;
  if (candidate_mag_Qrys.Overlaps(queriesToRun))
    candidate_mag = data_candidate_mag[i];

  if (qry.Overlaps(Q1) &&
      !(((candidate_ra BETWEEN L11 and H11) &&
         (candidate_dec BETWEEN L12 and H12) &&
         (candidate_b_image > 0.7) &&
         (candidate_a_image < 3.0 || candidate_mag < 15.0)))
    qry.Difference(Q1);

  if (qry.Overlaps(Q2) &&
      !(((candidate_ra BETWEEN L21 and H21) &&
         (candidate_dec BETWEEN L22 and H22) &&
         (candidate_b_image > 0.7) &&
         (candidate_a_image < 3.0 || candidate_mag < 15.0)))
    qry.Difference(Q2);

    queries[i] = qry;
}
```

Complex Processing Beyond SQL

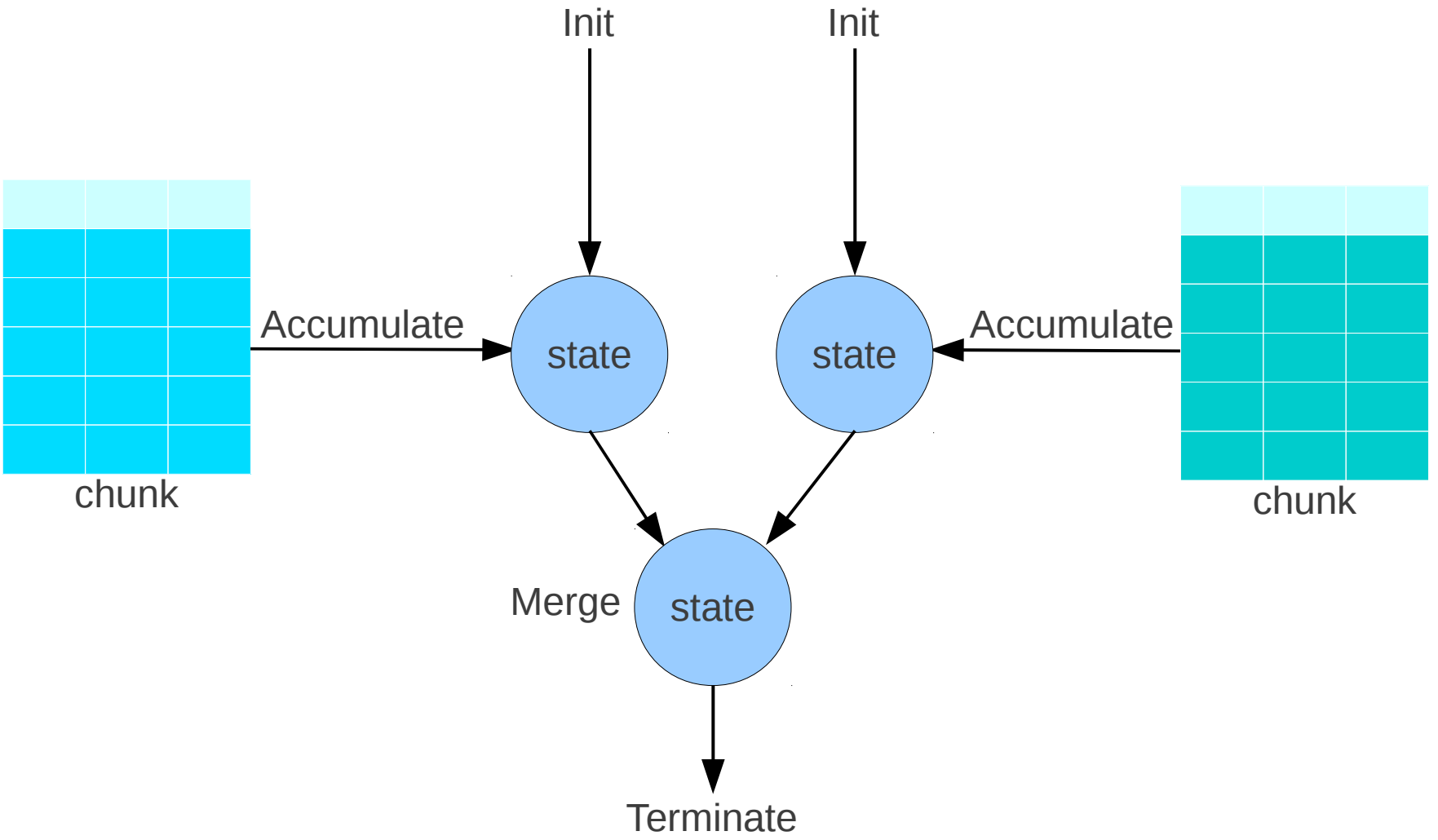
```
SELECT c.id, c.sub_id, c.mag,
  c.mag_ref, c.sym,
  c.xint_new, c.yint_new,
  c.a_image, c.b_image,
  ((c.xint_new - Cx)^2 +
   (c.yint_new - Cy)^2) ^0.5 AS dist,
  s.good_pix_area, rbc.realbogus
FROM subtraction s JOIN
  candidate c ON (c1.sub_id = s1.id)
  JOIN rb_classifier rbc ON
  (rbc.sub_id = c.sub_id AND
   rbc.candidate_id = c1.id)
WHERE
  c.sub_id = C AND c.pos_sub = 'T'
ORDER BY
  dist limit 20
```

+

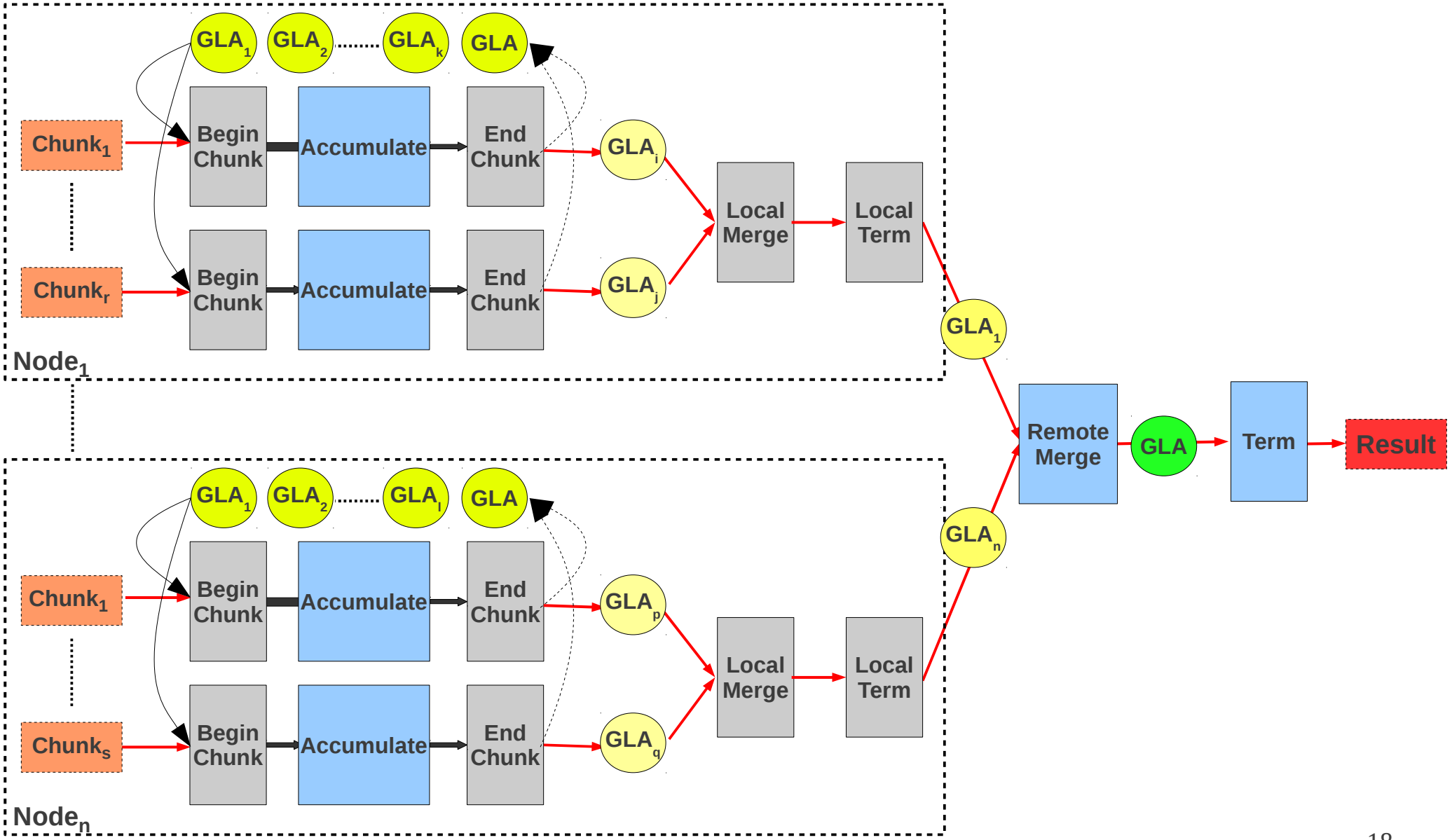
Application code



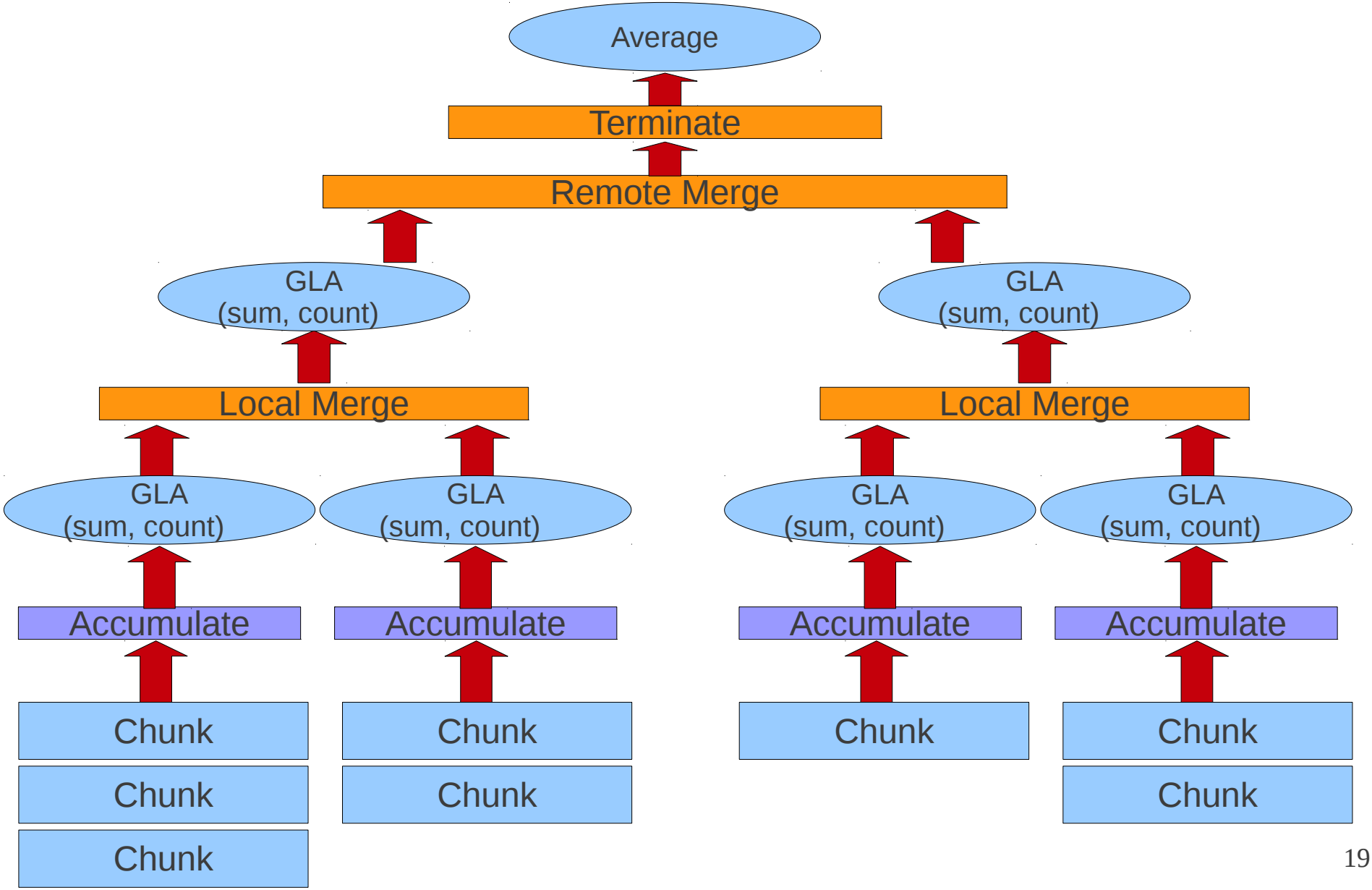
User-Defined Aggregate (UDA)



Generalized Linear Aggregates (GLA)



GLA Average Example



GLA Waypoint & GLA Manager

Waypoint

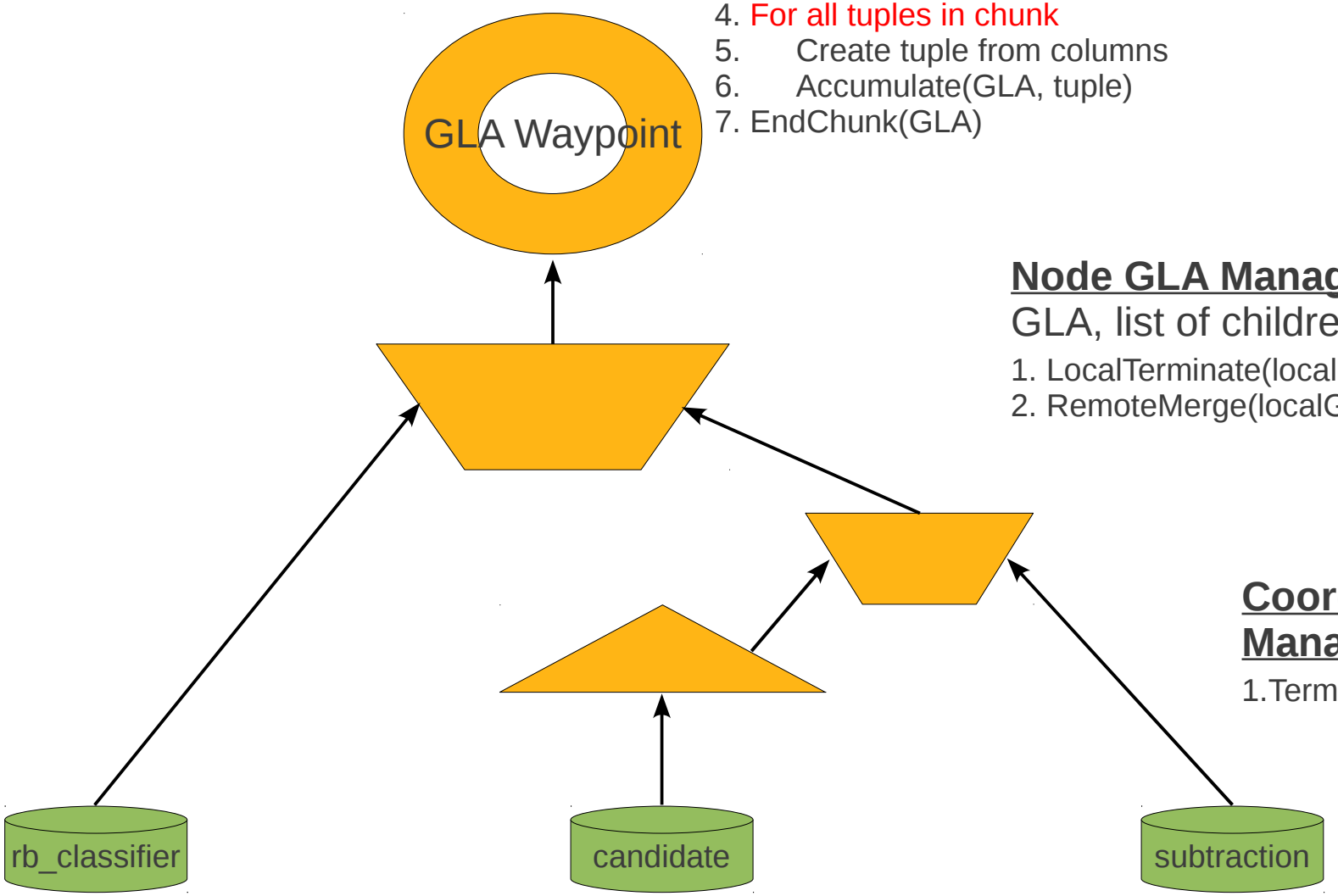
State: list of GLAs; some of them are out for execution in a work unit; some are in waypoint; a single GLA is produced in the end

ProcessChunk work unit
(chunk, GLA)

1. Init(GLA)
2. Extract columns from chunk
3. BeginChunk(GLA)
4. For all tuples in chunk
5. Create tuple from columns
6. Accumulate(GLA, tuple)
7. EndChunk(GLA)

MergeGLA work unit
(GLA_1, GLA_2)

1. LocalMerge(GLA_1, GLA_2)



Node GLA Manager (local GLA, list of children GLAs)

1. LocalTerminate(localGLA)
2. RemoteMerge(localGLA, children GLAs)

Coordinator GLA Manager (final GLA)

1.Terminate(final GLA)

No Problem Anymore

- October 10, 2011: 2,997 images; 1,939,059 candidates (647 per image); 40,087 identified; 5,600 non-singleton
- Raw data: 160 GB
- Intel Core2Quad CPU @ 2.66 GHz; 4 GB memory; 1 TB disk with sequential I/O throughput of 100 MB/s; Ubuntu SMP 10.10 64-bit

Phase	PostgreSQL	PostgreSQL + indexes	GLADE
Data ingestion	59 sec	**1 hour 8 sec**	3 sec
Identification	45 sec	4.67 sec	0.92 sec
Pruning	**607 hours**	15 min 39 sec	18 min
Contextual realbogus	68 hours	0.79 sec	1 min 30 sec
Total	**675 hours**	**1 hour 16 min**	**19 min 34 sec**

Questions ???