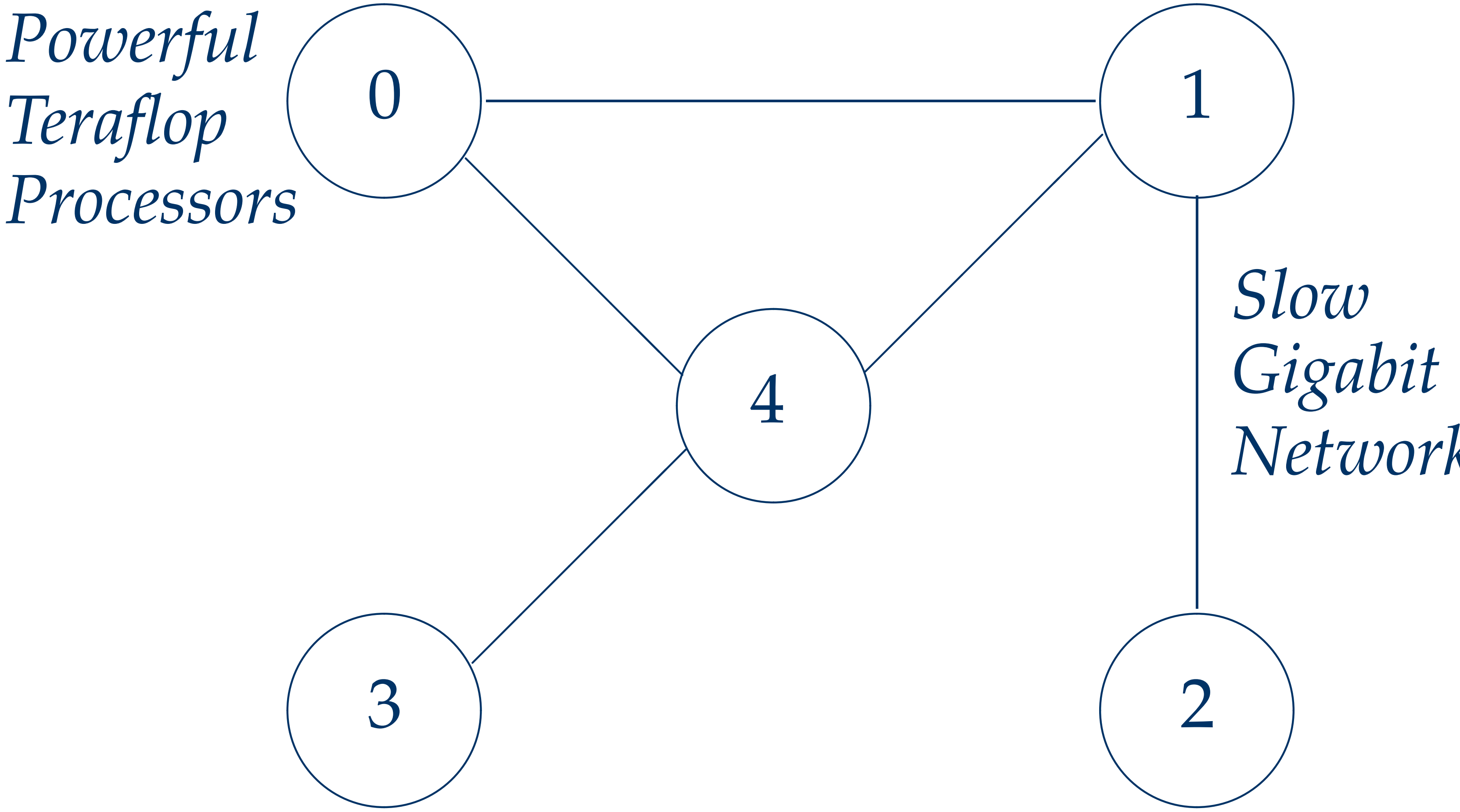
PCA for Distributed Data Sets

Raymond H. Chan
Department of Mathematics
The Chinese University of Hong Kong

Joint work with
Franklin Luk (RPI) and Z.-J. Bai (CUHK)

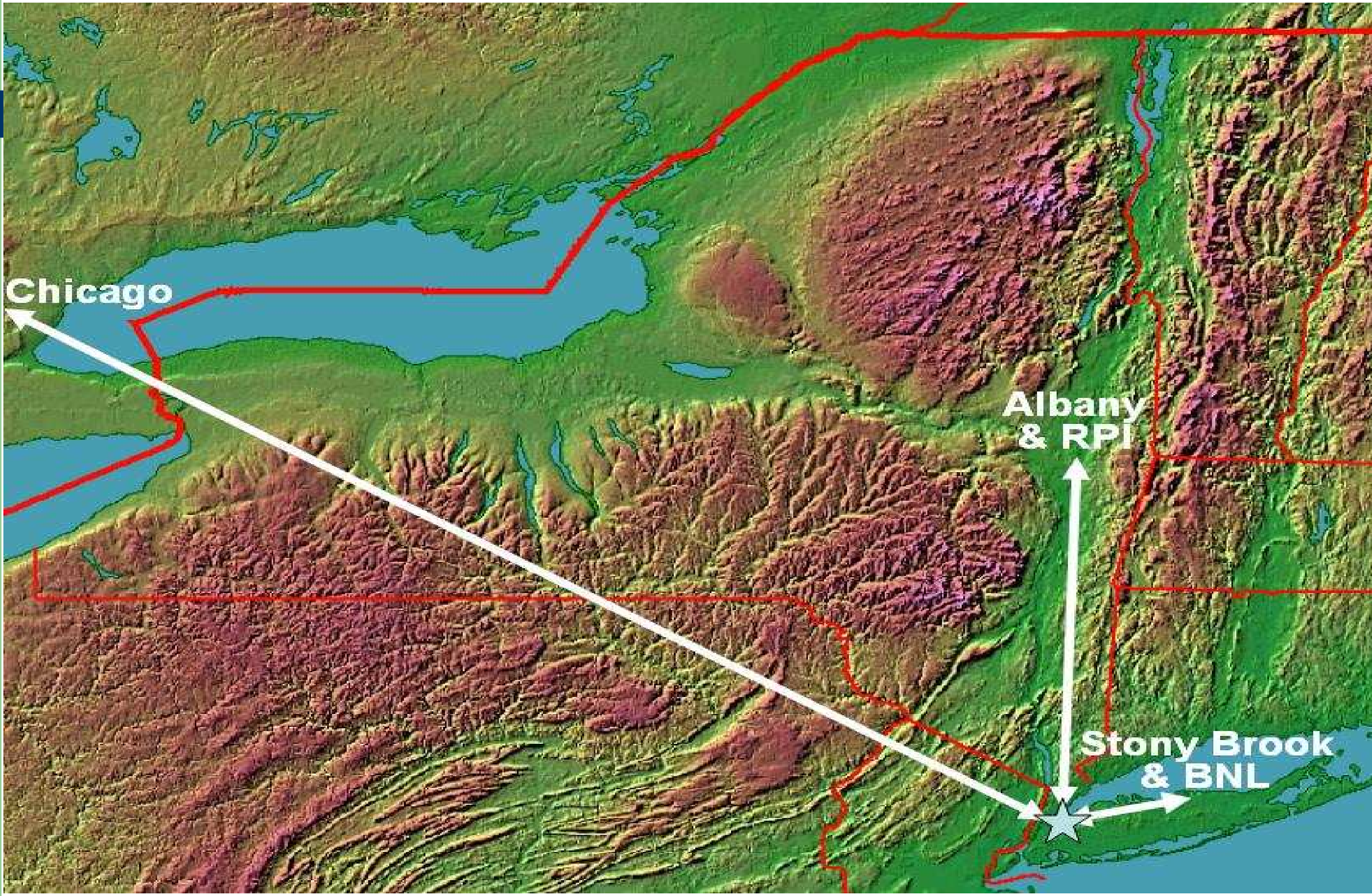
Grid

Powerful processors with relatively slow links.



New York State TeraGrid Initiative

- State University of NY at Albany
- Rensselaer Polytechnic Institute
- Brookhaven National Laboratory
- State University of NY at Stony Brook
- in partnership with IBM and NYSERNet


Chicago
Albany
& RPI
Stony Brook
& BNL

Grid Topology

- Four sites, scalable grid architecture
- 10 to 20 Gb/sec connection
- 6TF Processors

$$\frac{\text{Computation}}{\text{Communication}} = \frac{6 \times 10^{12}}{0.3 \times 10^{9}} = 20,000!$$

Communication Bottleneck:

computation done locally.

Principal Component Analysis

Let X be an $n \times p$ data matrix, where $n \gg p$.

Data covariace matrix S is given by

$$nS = X^T(I - \tfrac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)X,$$

where $\mathbf{e}_n^T = (1, 1, \ldots, 1)$.

$$\begin{aligned} \text{PCA} \quad &\Longleftrightarrow \quad \text{Karhunen-Loève transform} \\ &\Longleftrightarrow \quad \text{Hotelling transform} \end{aligned}$$

PCA means

- get spectral decomposition of $n \cdot S$:

$$nS = V\Sigma^2 V^T.$$

- choose few largest eigenvalues and eigenvectors $\widetilde{V}$.
- form principal component vectors $X \cdot \widetilde{V}$.
- low-dimension representation of original dat

PCA involves only V and Σ

Since $(I - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)$ is symmetric and idempotent,

$$\begin{aligned} nS &= X^T(I - \tfrac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)X \\ &= X^T(I - \tfrac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)(I - \tfrac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)X \end{aligned}$$

Σ and V can be obtained from SVD of:

$$(I - \tfrac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)X = U\Sigma V^T.$$

Low-dim'l representation $X \cdot \widetilde{V}$ can still be done.

Distributed PCA

Big data matrix X: $n \approx 10^{12}$.

E.g. visualization, data mining.

Problem:

Data are distributed amongst s processors.

Can we find Σ and V without moving X across processors?

Data among s processors

Denote

$$X = \left.\begin{pmatrix} X_0 \\ X_1 \\ \vdots \\ X_{s-1} \end{pmatrix}\right\} \quad n = \sum_{i=0}^{s-1} n_i,$$

where X_i is $n_i \times p$, resides on processor i.

Typical: $n_i \approx 10^{12}$ and $p \approx 10^3$.

Aim

- Compute PCA of X without moving the data matrix X_i.
- Move $O(p^\alpha)$ data across processors instead of $O(n_i)$.

Distributed PCA by Qu et al.

1. At processor i, calculate local PCA using SVD

$$(I - \frac{1}{n_i}\mathbf{e}_{n_i}\mathbf{e}_{n_i}^T)X_i = U_i\Sigma_iV_i^T.$$

Say matrix has numerical rank k_i.

Send $\bar{\mathbf{x}}_i$ (column sum of X_i), and k_i largest principal components $\widehat{\Sigma}_i$ and $\widehat{V}_i$ to central processor.

Communication costs = $O(pk_i)$.

2. At central processor: Assemble $p \times p$ covariance matrix and find its PCA

$$\begin{aligned} n\widehat{S} &= \sum_{i=0}^{s-1} \widehat{V}_i \widehat{\Sigma}_i^2 \widehat{V}_i^T + \sum_{i=0}^{s-1} n_i(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T \\ &= V\Sigma^2 V^T. \end{aligned}$$

Broadcast $\widetilde{V}$, the first k columns of V.

Communication costs = $O(pk)$.

3. Calculate principal component vectors at processor i:
$$X_i \widetilde{V}.$$

Analysis of Qu's Approach

Advantage: Reduce communication costs:

$$O(pn) \longrightarrow O\left(p\left(\sum_{i=0}^{s-1} k_i\right)\right).$$

Disadvantages:

- Local SVD's introduce approximation errors.
- Central processor becomes bottleneck for communications and computation.

Luk’s Algorithms

Replace SVD by QR

1a. At processor i

Calculate QR decomposition of $(I - \frac{1}{n_i}\mathbf{e}_{n_i}\mathbf{e}_{n_i}^T)X_i$:

$$(I - \frac{1}{n_i}\mathbf{e}_{n_i}\mathbf{e}_{n_i}^T)X_i = Q_i^{(0)}R_i^{(0)},$$

where $R_i^{(0)}$ is $p \times p$.

Send n_i and $\bar{\mathbf{x}}_i$ to central processor.

If $i \geq s/2$, send $R_i^{(0)}$ to processor $i - s/2$.

No need to send $Q_i^{(0)}$.

1b. At processor $i < s/2$

Calculate QR decomposition

$$\begin{pmatrix} R_i^{(0)} \\ R_{i+s/2}^{(0)} \end{pmatrix} = Q_i^{(1)} R_i^{(1)},$$

where $R_i^{(1)}$ is $p \times p$. Equals to QRD of

$$(I - \frac{1}{n_i}\mathbf{e}_{n_i}\mathbf{e}_{n_i}^T)X_i \quad \text{and} \quad (I - \frac{1}{n_{i+s/2}}\mathbf{e}_{n_{i+s/2}}\mathbf{e}_{n_{i+s/2}}^T)X_{i+s}$$

If $i \geq s/4$, send $R_i^{(1)}$ to processor $i - s/4$.

No need to send $Q_i^{(1)}$.

1c. At processor $i < s/4$

Calculate QR decomposition

$$\begin{pmatrix} R_i^{(1)} \\ R_{i+s/4}^{(1)} \end{pmatrix} = Q_i^{(2)} R_i^{(2)},$$

where $R_i^{(2)}$ is $p \times p$.

If $i \geq s/8$, send $R_i^{(2)}$ to processor $i - s/8$.

No need to send $Q_i^{(2)}$.

1d. Eventually, at processor 0

Calculate QR decomposition of

$$\begin{pmatrix} R_0^{(l-1)} \\ R_1^{(l-1)} \end{pmatrix} = Q_0^{(l)} R_0^{(l)},$$

where $l = \lceil \log_2 s \rceil$.

Send $R_0^{(l)}$ to central processor.

No need to send $Q_0^{(l)}$.

Main Results

- Total communication costs = $O(\lceil \log_2 s \rceil p^2)$.
- The covariance matrix

$$nS = X^T(I - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T)X,$$

is given by:

$$nS = {R_0^{(\ell)}}^T R_0^{(\ell)} + \sum_{i=0}^{s-1} n_i(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T.$$

2. At central processor

Assemble $(s+p) \times p$ data matrix:

$$Z = \begin{pmatrix} \sqrt{n_0}(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}})^T \\ \sqrt{n_1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ \sqrt{n_{s-1}}(\bar{\mathbf{x}}_{s-1} - \bar{\mathbf{x}})^T \\ R_0^{(l)} \end{pmatrix}.$$

Notice that: $nS = Z^T Z$.

2. At central processor

Compute SVD: $Z = U\Sigma V^T$ (after triangulation).

Say Z has numerical rank k.

Broadcast $\bar{\mathbf{x}}$ and $\widetilde{V}$, first k columns of V.

Communication costs = $O(pk)$.

3. At processor i

Calculate principal component vectors:

$$X_i \widetilde{V}.$$

Analysis of Luk’s Algorithm

Advantage over Qu’s Approach:

- Communication costs on PCA:

$$O\left(p\left(\sum_{i=0}^{s-1} k_i\right)\right) \longrightarrow O(p^2 \lceil \log_2 s \rceil),$$

- No local PCA approximation errors.
- Less congestion in central processor for communications and computation.
- Work directly with data matrices.

Data Updating

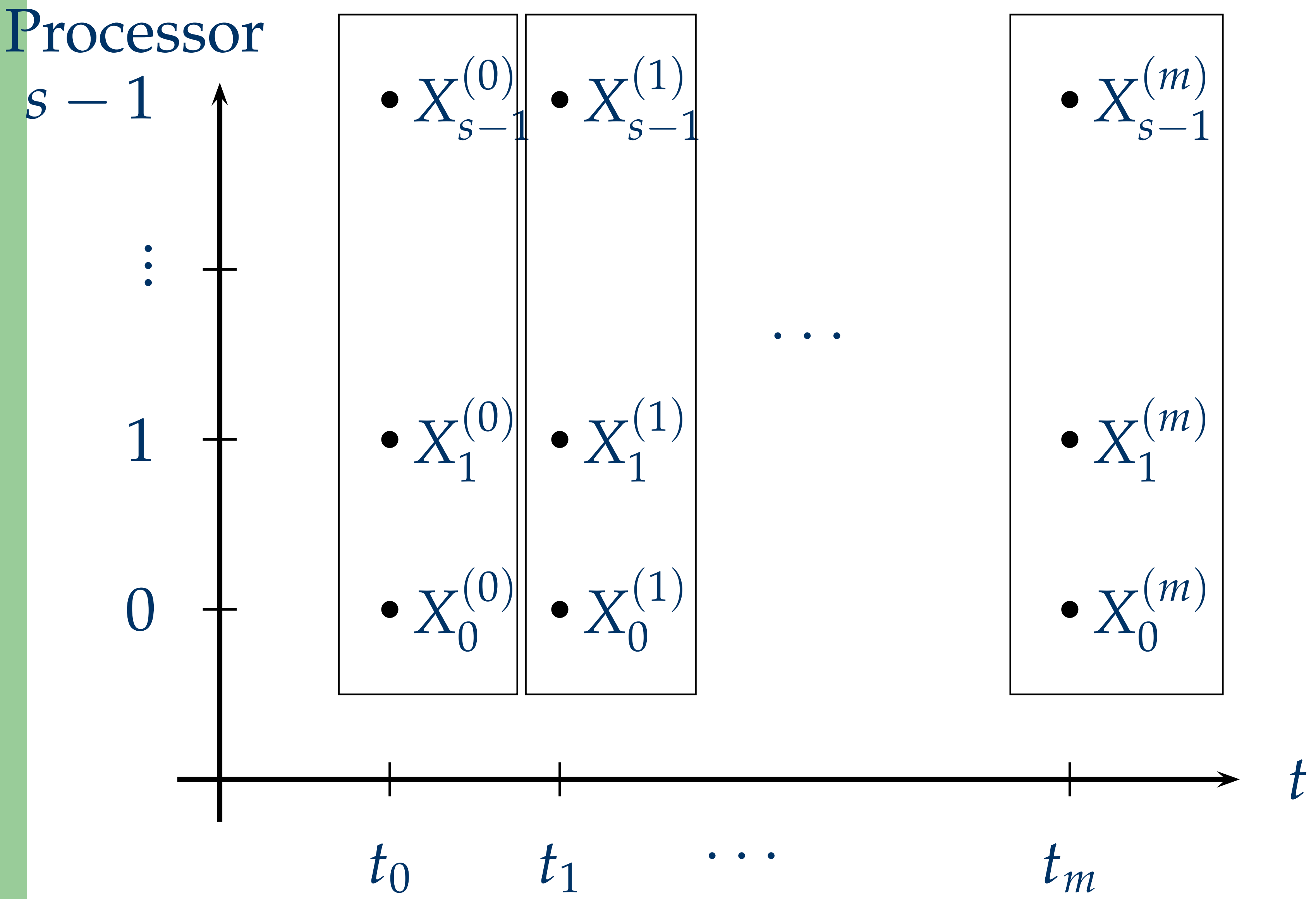
Assume global synchronization at $t_0, t_1, \ldots, t_k$, i.e. at $[t_{k-1}, t_k]$, new data are added to $X_i^{(k)}$ on processor i.

Aim:

Find the PCA for the new extended matrix, without moving $X_i^{(k)}$ across processors.

$\mathbb{X}^{(m)}$

$X^{(0)} \quad X^{(1)} \quad \cdots \quad X^{(m)}$



At time t_k

Let

$$X^{(k)} = \left.\begin{pmatrix} X_0^{(k)} \\ X_1^{(k)} \\ \vdots \\ X_{s-1}^{(k)} \end{pmatrix}\right\} \quad n^{(k)} = \sum_{i=0}^{s-1} n_i^{(k)},$$

where $X_i^{(k)}$ is $n_i^{(k)} \times p$.

Assume PCA of original matrix $X^{(0)} = X$ is available by Luk's algorithm.

Global Data Matrix at t_m

Denote

$$\mathbb{X}^{(m)} = \left.\begin{pmatrix} X^{(0)} \\ X^{(1)} \\ \vdots \\ X^{(s-1)} \end{pmatrix}\right\} \quad g(m) = \sum_{i=0}^{m} n^{(k)}.$$

Aim: Find PCA for its covariance matrix:

$$g(m)\cdot \mathbb{S}_{g(m)} = \mathbb{X}^{(m)^T}(I - \frac{1}{g(m)}\mathbf{e}_{g(m)}\mathbf{e}_{g(m)}^T)\mathbb{X}^{(m)}.$$

Our Theorem

Let

$$n^{(k)} \cdot S_k = X^{(k)^T}(I - \frac{1}{n^{(k)}}\mathbf{e}_{n^{(k)}}\mathbf{e}^T_{n^{(k)}})X^{(k)}.$$

Then

$$g(m)\mathbb{S}_{g(m)} = \sum_{k=0}^{m} n^{(k)} S_k$$

$$+ \sum_{k=1}^{m} \frac{g(k-1)n^{(k)}}{g(k)} (\bar{\mathbf{x}}_{g(k-1)} - \bar{\mathbf{x}}_{n^{(k)}})(\bar{\mathbf{x}}_{g(k-1)} - \bar{\mathbf{x}}_{n^{(k)}})$$

Explanation

PCA of update data matrix $X^{(k)}$ can be obtained b
Luk's algorithm, i.e. $n^{(k)}S_k = R_k^T R_k$. Then

$$g(m)\mathbb{S}_{g(m)} = \sum_{k=0}^{m} R_k^T R_k$$

$$+ \sum_{k=1}^{m} \frac{g(k-1)n^{(k)}}{g(k)} \left(\bar{\mathbf{x}}_{g(k-1)} - \bar{\mathbf{x}}_{n^{(k)}}\right)\left(\bar{\mathbf{x}}_{g(k-1)} - \bar{\mathbf{x}}_{n^{(k)}}\right)$$

Assemble them to construct global PCA for $\mathbb{X}^{(m)}$.

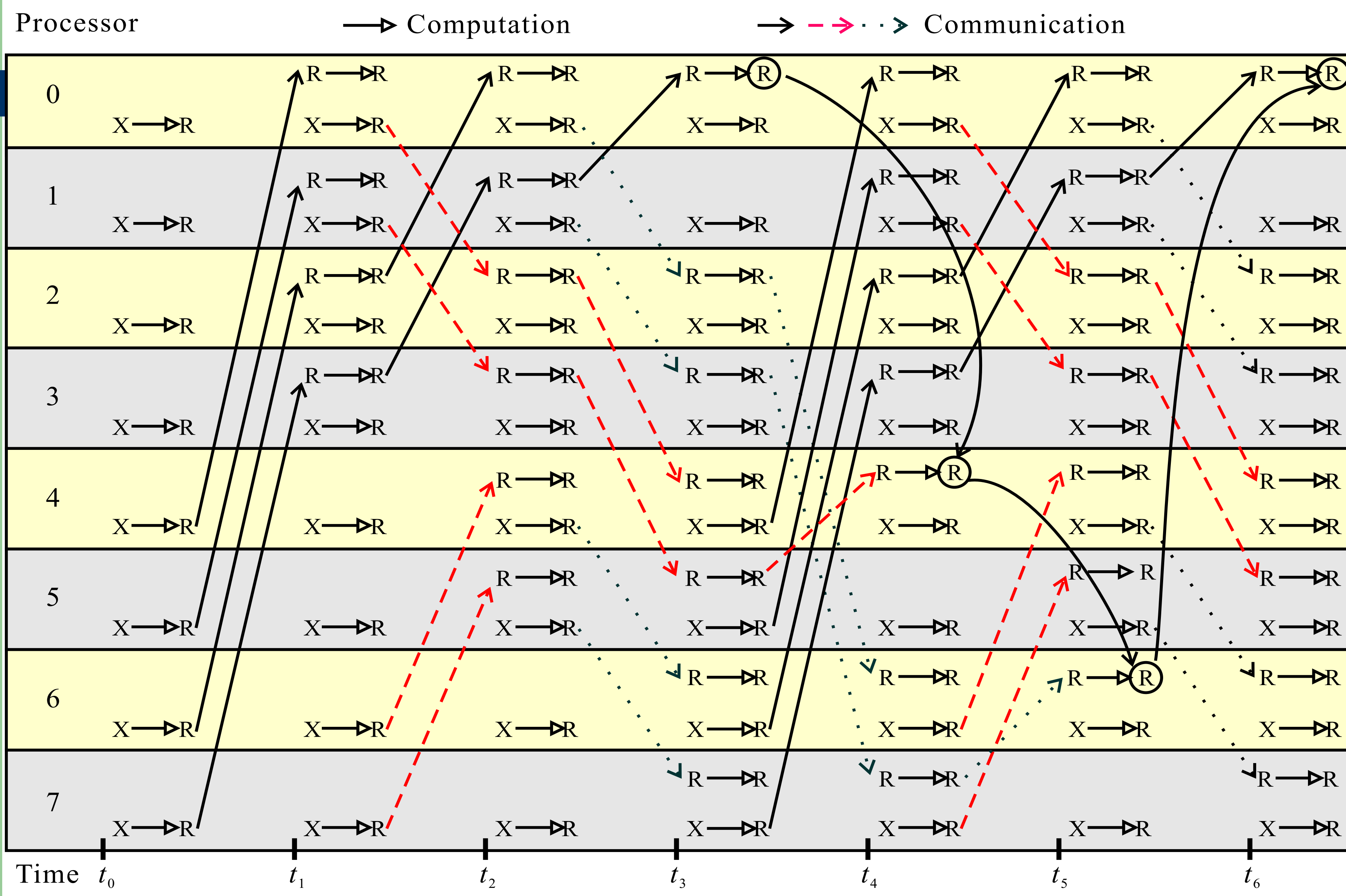
Analysis of Our Algorithm

- Global PCA can be computed without movin $X^{(k)}$.
- Communication costs still $O(p^2\lceil \log_2 s \rceil)$,
- No local PCA approximation errors.
- Work directly with data matrices and update matrices.
- Load balancing for communications and computation.

Load Balancing

Let $s = 2^\ell$. We can allocate all processors to do the
QR factorizations such that:

- PCA of $\mathbb{X}^{(k)} \leftarrow$ PCA of $\mathbb{X}^{(k-1)}$ $+$ R factor of $X^{(k)}$.
- PCA of $\mathbb{X}^{(k)}$ obtained in $t_{k+\ell}$.
- The procedure is periodic with period ℓ.
- Well-balanced among the processors.


Processor
Computation
Communication
0
1
2
3
4
5
6
7
X
R
Time
t_0
t_1
t_2
t_3
t_4
t_5
t_6