Online Balanced Repartitioning

Chen Avin

Ben Gurion University of the Negev

Joint work with Andreas Loukas, Maciej Pacut & Stefan Schmid
Motivation
Motivation

- Graph partitioning problems
Motivation

- Graph partitioning problems
- $\ell$ clusters, each of size $k$
Motivation

• Graph partitioning problems

• \( \ell \) clusters, each of size \( k \)
Motivation

- Graph partitioning problems
- $\ell$ clusters, each of size $k$
- Online graph re-partitioning
Motivation

- Graph partitioning problems
- $\ell$ clusters, each of size $k$
- Online graph re-partitioning
- Edges are updated
Motivation

- Graph partitioning problems
  - $\ell$ clusters, each of size $k$
- Online graph re-partitioning
  - Edges are updated
  - Clustering is updated
Motivation

- Graph partitioning problems
  - $\ell$ clusters, each of size $k$
- Online graph re-partitioning
  - Edges are updated
  - Clustering is updated
  - At a cost
Motivation
Motivation

• Practical motivation
  • Data centres
  • Reduce network traffic
Motivation

• Practical motivation
  • Data centres
  • Reduce network traffic
  • Clusters as servers (static)
  • Nodes as VMs (can move)
  • $k << \ell$

Motivation

• Practical motivation
  • Data centres
  • Reduce network traffic
  • Clusters as servers (static)
  • Nodes as VMs (can move)
  • $k << \ell$

• Traffic-Aware Networking
Overview

• Motivation
• Model and Problem definition
• Examples
• Some results
• Future work and open questions
Model
Model

- Balanced RePartitioning (BRP)
Model

- **Balanced RePartitioning (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size $k$
Model

- **Balanced RePartitioning (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size $k$
  - (online) pairwise communication requests
    \[ \sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots \]
Model

- **Balanced RePartitioning (BRP)**

- Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size $k$

- (online) pairwise communication requests

  $$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

- Serving **costs** for $\sigma_t = \{u, v\}$:
Model

- **Balanced RePartitioning (BRP)**
  - Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size $k$
  - (online) pairwise communication requests

$$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$$

- Serving costs for $\sigma_t = \{u, v\}$:
  - **intra-cluster**: 0
Model

• **Balanced RePartitioning (BRP)**

  • Clusters \( C = \{C_1, \ldots, C_\ell\} \) each of size \( k \)

  • (online) pairwise communication requests

    \[ \sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots \] 

  • Serving **costs** for \( \sigma_t = \{u, v\} \):

    • **intra-cluster:** 0
    • **inter-cluster:** 1

Model

• **Balanced RePartitioning (BRP)**

  • Clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ each of size $k$

  • (online) pairwise communication requests

    $\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$

  • Serving **costs** for $\sigma_t = \{u, v\}$:

    • intra-cluster: 0
    • inter-cluster: 1
    • migration: $\alpha$
Problem Definition
Problem Definition

• The Cost of $\text{ALG}$

\[
\text{ALG}(\sigma) = \sum_{t=1}^{\left|\sigma\right|} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG})
\]

$\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots$
Problem Definition

- The Cost of \textbf{ALG}

\[
\text{ALG}(\sigma) = \sum_{t=1}^{|\sigma|} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG})
\]

\[\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots\]
Problem Definition

- The Cost of ALG

\[
ALG(\sigma) = \sum_{t=1}^{\left|\sigma\right|} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG})
\]

\(\sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots\)
Problem Definition

- The Cost of $\text{ALG}$

\[
\text{ALG}(\sigma) = \sum_{t=1}^{\left|\sigma\right|} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG})
\]

- What is the competitive ratio

\[
\rho(\text{ON}) = \max_{\sigma} \frac{\text{ON}(\sigma)}{\text{OFF}(\sigma)}
\]
Problem Definition

• The Cost of ALG

\[ ALG(\sigma) = \sum_{t=1}^{\mid \sigma \mid} \text{mig}(\sigma_t; \text{ALG}) + \text{com}(\sigma_t; \text{ALG}) \]

\( \sigma = \{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots \)

• What is the competitive ratio

\[ \rho(\text{ON}) = \max_{\sigma} \frac{\text{ON}(\sigma)}{\text{OFF}(\sigma)} \]

• w/o Augmentation
Example 1: $\ell = 2$
Example 1: \( \ell = 2 \)

- The static variant corresponds to the minimum bisection problem - hard, but approx
Example 1: $\ell = 2$

- The static variant corresponds to the minimum bisection problem - hard, but approx

- The dynamic case is a generalization of online paging

\[
\begin{array}{c|c}
\text{cache} & \text{disk} \\
\end{array}
\]
Example 1: $\ell = 2$

• The static variant corresponds to the minimum bisection problem - hard, but approx

• The dynamic case is a generalization of online paging

• Imply $k$ lower bound (deterministic)
Example 1: $\ell = 2$

- The static variant corresponds to the minimum bisection problem - hard, but approx
- The dynamic case is a generalization of online paging
- Imply $k$ lower bound (deterministic)
- With augmentation it’s different....

\[ \begin{array}{c}
\text{cache} \\
\text{disk}
\end{array} \]
Example II: $k = 2$
Example II: $k = 2$

- The iid variant corresponds to the maximum matching problem (minimum cut)
Example II: $k = 2$

- The iid variant corresponds to the maximum matching problem (minimum cut)
Example II: $k = 2$

- The iid variant corresponds to the maximum matching problem (minimum cut)

- A novel online version of maximum matching
Algo Guidelines

• Serve remotely or migrate ("rent or buy")?

• Where to migrate, and what?

• Which nodes to evict?
Related work

• Similar in spirit to many classical on-line problems:
  • ski rental, page and server migration, k-server, caching, bin packing

• However, does not fit to the online metrical task system scenario
  • both ends of the communication requests can move
  • every request only reveals partial and and limited information about the optimal configuration
  • large space

• Caching models \textit{with bypassing}
Results overview

- Bounds for deterministic algorithms
- $k=2$ - constant competitive bound
- Lower bound (with augmentation) - $\Omega(k)$
- Upper bound (with augmentation) - $O(k \log k)$
\[ k = 2 \]
\[ k = 2 \]

- We show a lower bound of 3-competitive
We show a lower bound of 3-competitive

No eviction problem 😊
We show a lower bound of 3-competitive

No eviction problem 😊

A greedy algorithm:
$k = 2$

- We show a lower bound of 3-competitive
- No eviction problem 😊
- A **greedy** algorithm:
  - When outside traffic $> 3\alpha$
$k = 2$

- We show a lower bound of 3-competitive
- No eviction problem 😊
- A **greedy** algorithm:
  - When outside traffic $> 3\alpha$
$k = 2$

- We show a lower bound of 3-competitive.
- No eviction problem 😊.
- A **greedy** algorithm:
  - When outside traffic $> 3\alpha$
  - Identify and migrate to best cluster
\[ k = 2 \]

- We show a lower bound of 3-competitive
- No eviction problem 😊
- A **greedy** algorithm:
  - When outside traffic > 3\(\alpha\)
  - Identify and migrate to best cluster
- An upper bound of 7\(\alpha\)
\[ k = 2 \]

- We show a lower bound of 3-competitive
- No eviction problem 😊
- A greedy algorithm:
  - When outside traffic > 3\( \alpha \)
  - Identify and migrate to best cluster
- An upper bound of 7\( \alpha \)
- no augmentation
Lower Bound
Lower Bound

• Any non trivial augmentation (all fit in one)
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an inter-cluster edge to ask
Lower Bound

- Any non trivial augmentation (all fit in one)
- A simple cycle like request sequence
- Always an inter-cluster edge to ask
Lower Bound

• Any non trivial augmentation (all fit in one)
• A simple cycle like request sequence
• Always an inter-cluster edge to ask
Lower Bound

• Any non trivial augmentation (all fit in one)
• A simple cycle like request sequence
• Always an inter-cluster edge to ask
• Leads to a lower bound $> k$
Upper Bound
Upper Bound

• C-REP algorithm (Component-based)
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation
Upper Bound

• C-REP algorithm (Component-based)

• 4 Augmentation
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation
Upper Bound

- C-REP algorithm (Component-based)
- 4 Augmentation

- **Theorem**: CREP is $O(k \log k)$ competitive.
Component based
Component based

• Communication components
Component based

• Communication components
Component based

- Communication components
Component based

- Communication components
Component based

- Communication components
Component based

- Communication components
Component based

• Communication components
Component based

- Communication components
Component based

- Communication components
Component based

- Communication components
Component based

- Communication components
- Merge while you can
Component based

- Communication components
- Merge while you can
- Small-to-large component
Component based

• Communication components
• Merge while you can
• Small-to-large component
• Only move once to a “new” cluster
Component based

- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster
- Component larger than $k$ we can safely charge OFF

“new”
Component based

• Communication components
• Merge while you can
• Small-to-large component
• Only move once to a “new” cluster
• Component larger than \( k \) we can safely charge OFF
• Epoch ends…. split cluster to singletons

“new”
Component based

- Communication components
- Merge while you can
- Small-to-large component
- Only move once to a “new” cluster
- Component larger than $k$ we can safely charge OFF
- Epoch ends…. split cluster to singletons
- Need to be careful on the condition to merge
To establish the base case, consider the first merge of nodes in merges that includes all the nodes in Property 2.1.

Proof: in this epoch, then:

Property 2. properties provide upper bounds for both kinds of costs for a single component: communication cost Upper bound on CREP's costs. component. ensures that each node is migrated at most log

CREP at least component, including the required reserved space without any evacuation, i.e., its spare space is

So indeed,

We start by observing that there always exists a cluster which can host the entire merged at least one newly created component. Reserve one additional space for each newly created component.

At any point in time, consider a component

\[ \text{reserved}(X) = \text{vol}(X) - |\phi| \]

Let \( \phi_0 = \bigcup_{\phi_i \in X} \phi_i \) and for all \( \phi_j \in \Phi \setminus X \) set \( w_{0j} = \sum_{\phi_i \in X} w_{ij} \).

If \( \text{reserved}(\phi) \geq \text{vol}(X) - |\phi| \)

Migrate \( \phi_0 \) to the cluster hosting \( \phi \)

Update \( \text{reserved}(\phi_0) = \text{reserved}(\phi) - (\text{vol}(X) - |\phi|) \)

else

Migrate \( \phi_0 \) to a cluster \( s \) with \( \text{spare}(s) \geq \min(k, 2|\phi_0|) \)

Set \( \text{reserved}(\phi_0) = \min(k - |\phi_0|, |\phi_0|) \)

End of a \( Y \)-epoch.

Let \( Y \) be the smallest components set with \( \text{vol}(Y) > k \) and \( \text{com}(Y) \geq \text{vol}(Y) \cdot \alpha \)

if \( Y \neq \emptyset \) then

Split every \( \phi_i \in Y \) into \( \phi_i \) singleton components and reset the weights of all edges involving at least one newly created component. Reserve one additional space for each newly created component. If necessary, migrate at most \( \text{vol}(Y)/2 + 1 \) singletons to clusters with spare space.

end if

end for
Algorithm 1 CREP with 4 Augmentation

1: Construct graph $G = (\Psi, E, w)$ with singleton components: one component per node. Set $w_{ij} = 0$ for all $\{v_i, v_j\} \in \binom{\Psi}{2}$. For each component $\phi_i$, reserve space $\text{reserve}(\phi_i) = 1$.

2: for each new request $\{u_t, v_t\}$ do
   \hspace{1cm} \triangleright \text{Keep track of communication cost.}
3: \hspace{2cm} Let $\phi_i = \Phi(u_t)$ and $\phi_j = \Phi(v_t)$ be the two components that communicated.
4: \hspace{2cm} if $\phi_i \neq \phi_j$ then
5: \hspace{3cm} $w_{ij} \leftarrow w_{ij} + 1$
6: \hspace{2cm} end if
    \hspace{1cm} \triangleright \text{Merge components.}
7: \hspace{2cm} Let $X$ be the largest cardinality set with $\text{vol}(X) \leq k$ and $\text{com}(X) \geq (|X| - 1) \cdot \alpha$
8: \hspace{2cm} if $|X| > 1$ then
9: \hspace{3cm} Let $\phi_0 = \bigcup_{\phi_i \in X} \phi_i$ and for all $\phi_j \in \Psi \setminus X$ set $w_{0j} = \sum_{\phi_i \in X} w_{ij}$
10: \hspace{2cm} Let $\phi \in X$ be the component having the largest reserved space.
11: \hspace{2cm} if $\text{reserved}(\phi) \geq \text{vol}(X) - |\phi|$ then
12: \hspace{3cm} Migrate $\phi_0$ to the cluster hosting $\phi$
13: \hspace{3cm} Update $\text{reserved}(\phi_0) = \text{reserve}(\phi) - (\text{vol}(X) - |\phi|)$
14: \hspace{2cm} else
15: \hspace{3cm} Migrate $\phi_0$ to a cluster with spare $s \geq \min(k, 2|\phi_0|)$
16: \hspace{3cm} Set $\text{reserved}(\phi_0) = \min(k - |\phi_0|, |\phi_0|)$
17: \hspace{2cm} end if
18: \hspace{2cm} end if
    \hspace{1cm} \triangleright \text{End of a } Y \text{-epoch.}
19: \hspace{2cm} Let $Y$ be the smallest components set with $\text{vol}(Y) > k$ and $\text{com}(Y) \geq \text{vol}(Y) \cdot \alpha$
20: \hspace{2cm} if $Y \neq 1$ then
21: \hspace{3cm} Split every $\phi_i \in Y$ into $\phi_i$ singleton components and reset the weights of all edges involving at least one newly created component. Reserve one additional space for each newly created component.
22: \hspace{3cm} If necessary, migrate at most $\text{vol}(Y)/2 + 1$ singletons to clusters with spare space.
22: \hspace{2cm} end if
23: end for
Open questions

• Randomize algorithms (lower and upper bounds)
  • Some initial results

• A better network model than one-switch network

• Similar models that fits better in practice (e.g., MapReduce. etc.)

• Open Postdoc position (Beer-Sheva and Berlin) to work on these problems… feel free to talk to me.
Thank you!