# Giraph Unchained: BAP Execution in Pregel-like Graph Processing Systems

Minyang Han and Khuzaima Daudjee

University of Waterloo

VLDB 2015

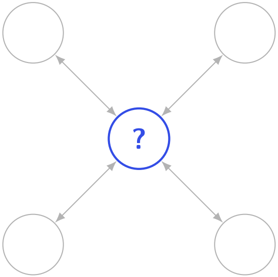
## Graph Processing Systems

We focus on **vertex-centric** graph processing systems.

## Graph Processing Systems

We focus on **vertex-centric** graph processing systems.

- "Think like a vertex":

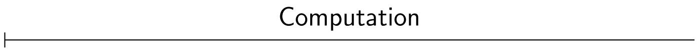


## Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.

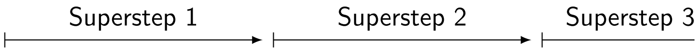
## Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.



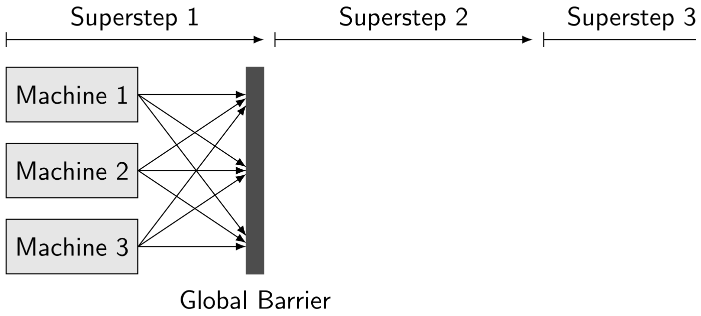
## Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.



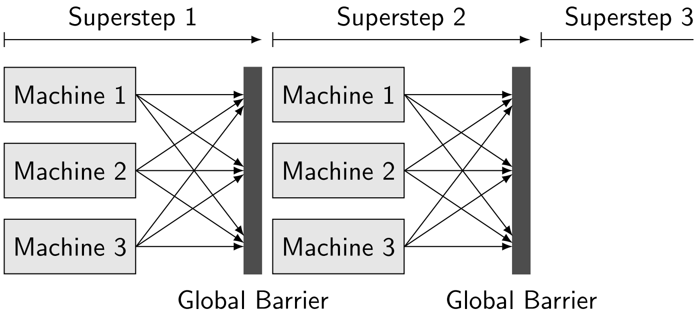
# Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.



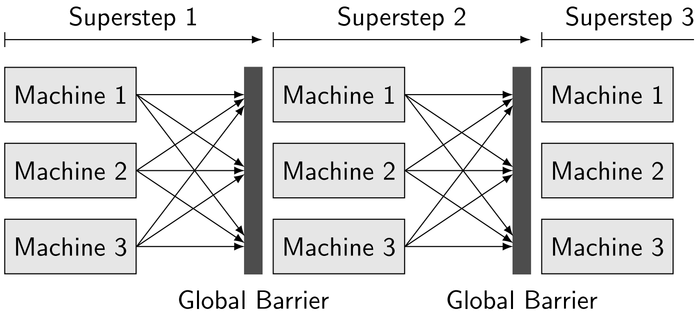
## Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.

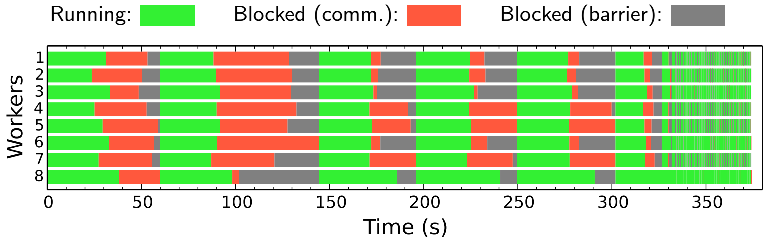


## Pregel-like Graph Processing Systems

Pregel-like systems are vertex-centric, **BSP** programs.



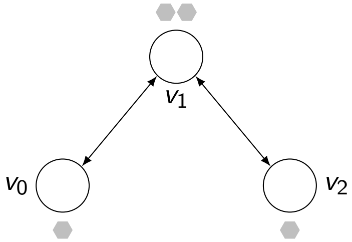
## Motivation



BSP execution: 46% of total time is spent blocked!

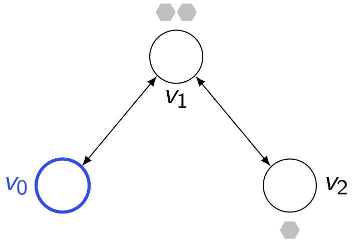
## Performance Issues of BSP

1. Stale messages:



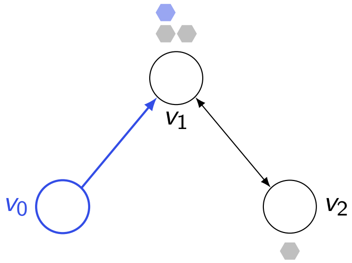
## Performance Issues of BSP

1. Stale messages:



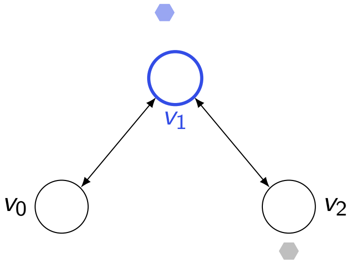
## Performance Issues of BSP

1. Stale messages:



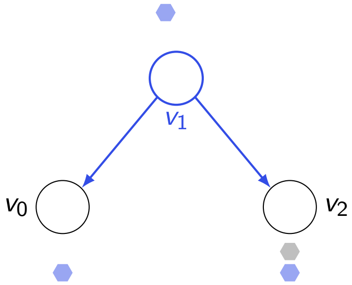
## Performance Issues of BSP

1. Stale messages:



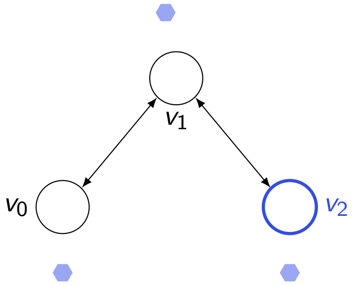
## Performance Issues of BSP

1. Stale messages:



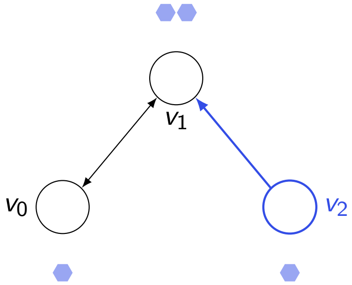
## Performance Issues of BSP

1. Stale messages:



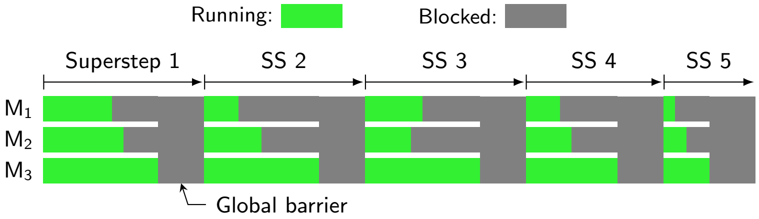
## Performance Issues of BSP

1. Stale messages:



## Performance Issues of BSP

2. Expensive global barriers and *stragglers*:



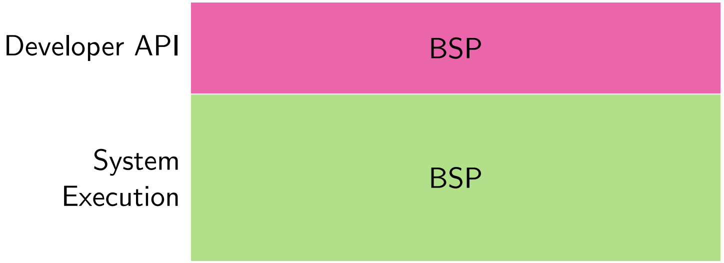
## Why use BSP?

Significant benefits for algorithm developers:

- Easy to design and reason about algorithms.
- Simplifies debugging.
- Many existing BSP algorithms.

# The Big Picture

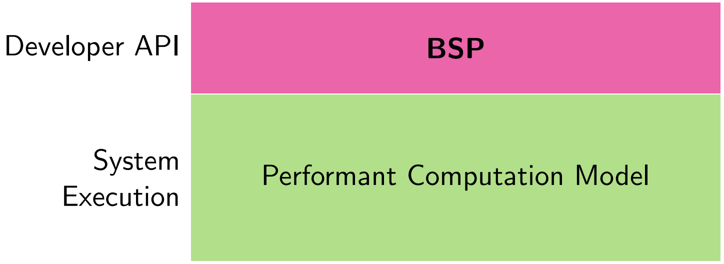
Pregel-like graph processing systems:



E.g., Pregel, Giraph

## Best of Both Worlds

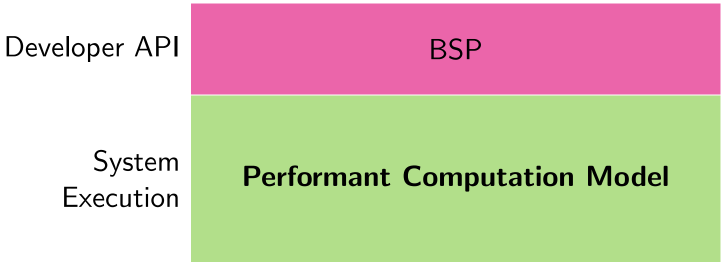
Retain support for BSP developer interface:



A performant graph processing system.

# Best of Both Worlds

Retain support for BSP developer interface:



A performant graph processing system.

## Summary

**Objective 1:** Allow vertices to use more recent messages.

**Objective 2:** Minimize global barriers and stragglers.

**Objective 3:** Improve performance while retaining a BSP developer interface.

## Overview

## Overview

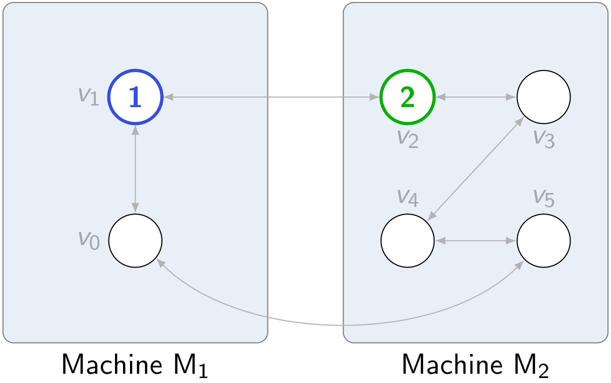
# Barrierless Asynchronous Parallel (BAP)

| **Computation Model** | **Reduce Stale Messages (O1)** | **Minimize Barriers (O2)** | **BSP Algorithm Support (O3)** | **Scalable** |
|---|---|---|---|---|
| BSP/Sync GAS | ✗ | ✗ | ✓/✗ | ✓ |
| AP | ✓ | ✗ | ✗ | ✓ |
| Async GAS | ✓ | ✓ | ✗ | ✗ |
| **BAP** | ✓ | ✓ | ✓ | ✓ |

Asynchronous Parallel: asynchronous extension of BSP.

Gather, Apply, Scatter: computation model used in GraphLab.

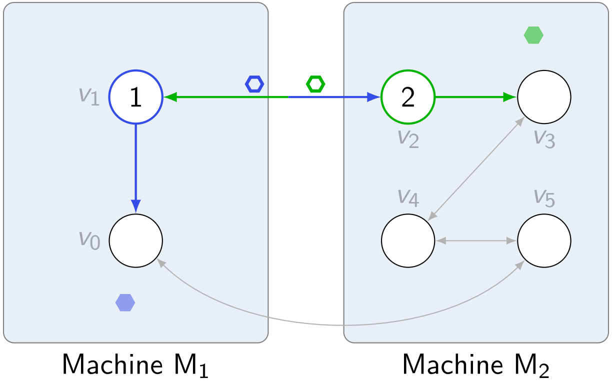
# BSP on WCC Algorithm



$M_1$ 🟩

$M_2$ 🟩

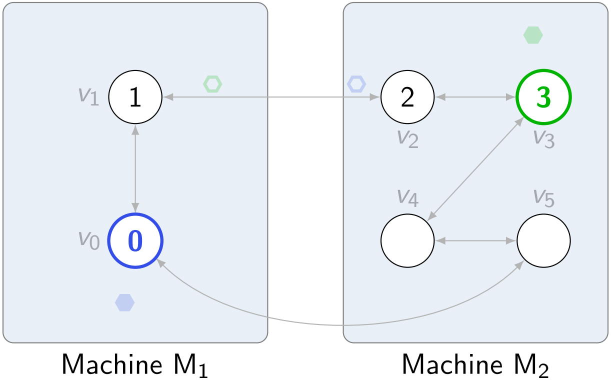
## BSP on WCC Algorithm



$M_1$ 🟩

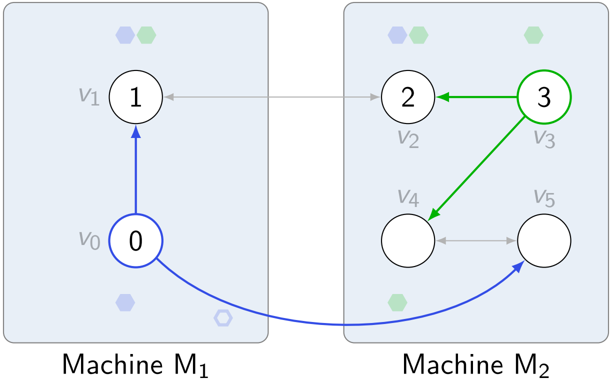
$M_2$ 🟩

# BSP on WCC Algorithm



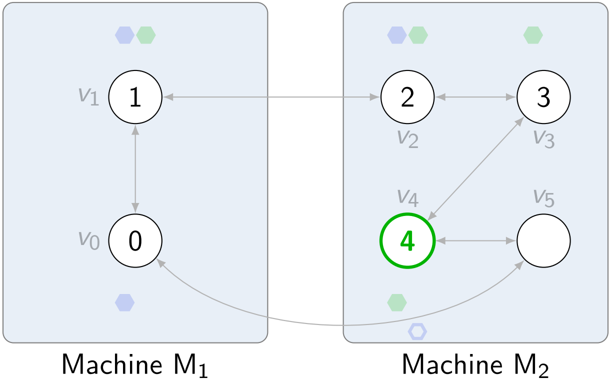
| | |
|---|---|
| $M_1$ | 🟩 |
| $M_2$ | 🟩 |

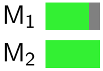
# BSP on WCC Algorithm



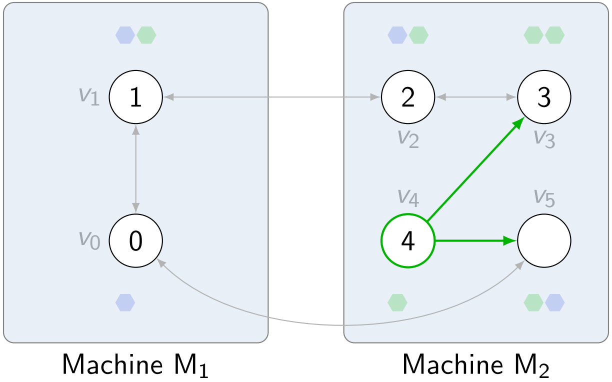
| | |
|---|---|
| $M_1$ | 🟩 |
| $M_2$ | 🟩 |

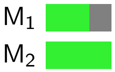
# BSP on WCC Algorithm



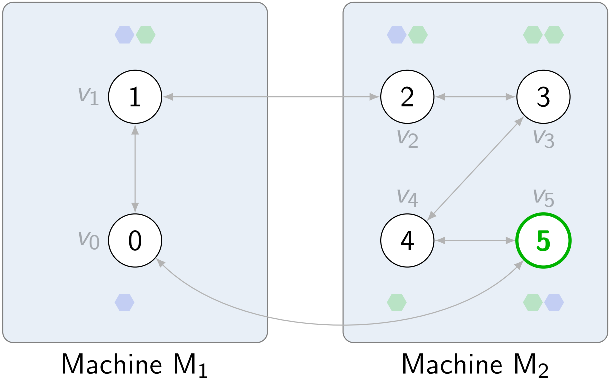


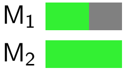
# BSP on WCC Algorithm



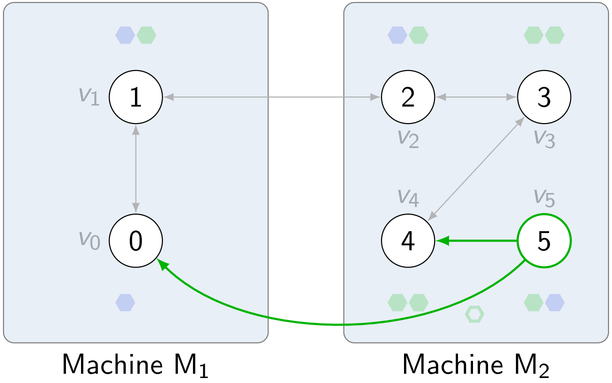


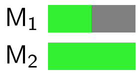
## BSP on WCC Algorithm



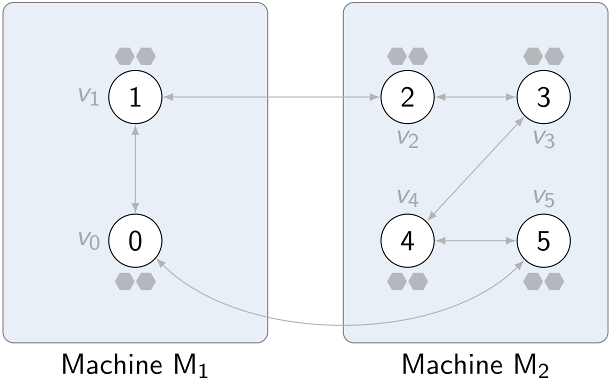


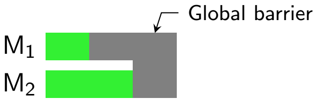
# BSP on WCC Algorithm



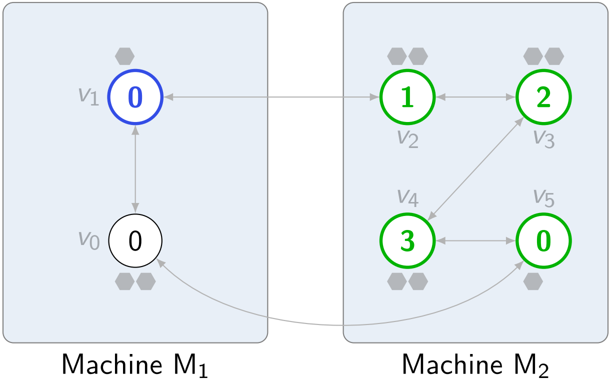


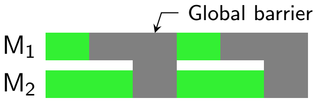
# BSP on WCC Algorithm



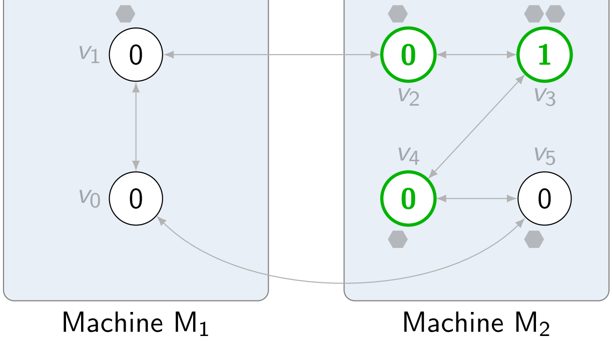


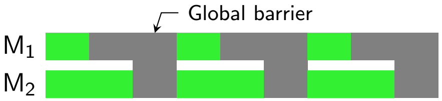
## BSP on WCC Algorithm



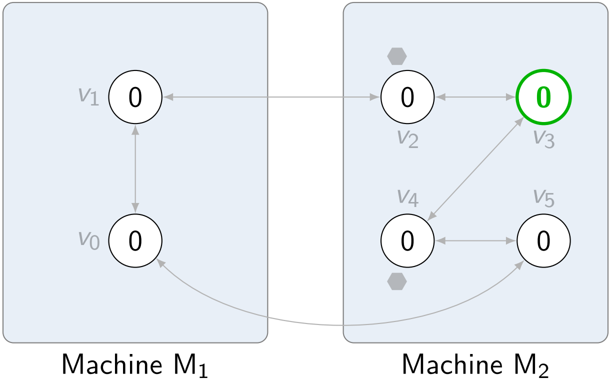


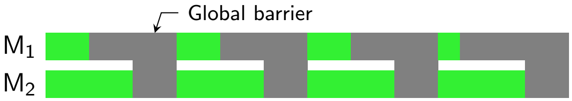
## BSP on WCC Algorithm



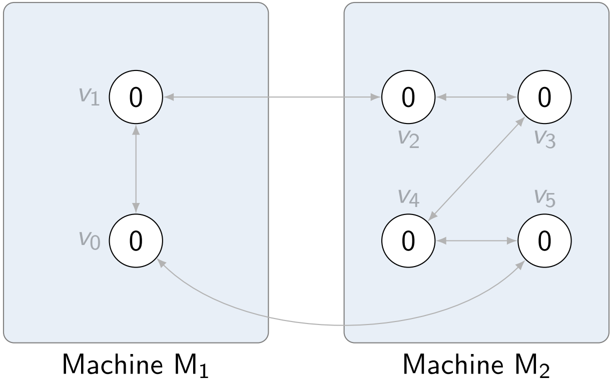


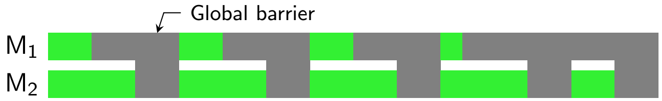
## BSP on WCC Algorithm





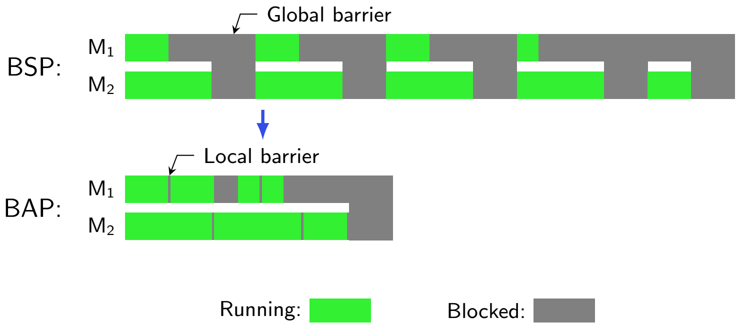
## BSP on WCC Algorithm



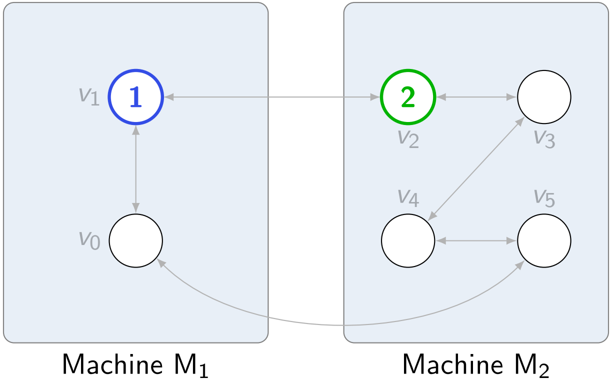


## Intuition for BAP (O1 and O2)

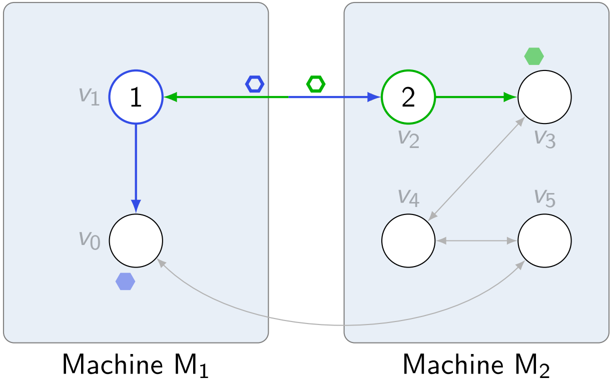
- Reduce stale messages (**O1**) by adding *asynchrony*.
- Minimize global barriers (**O2**) by using *local barriers*.



# BAP on WCC Algorithm



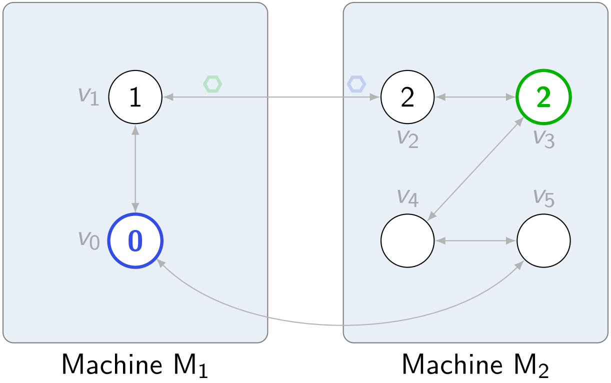
# BAP on WCC Algorithm



$M_1$ 🟩

$M_2$ 🟩

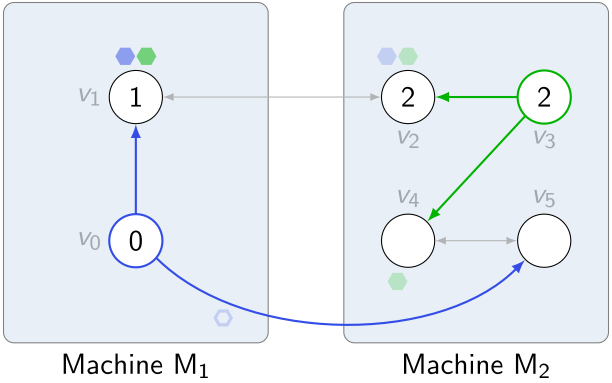
# BAP on WCC Algorithm

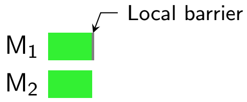


$M_1$ 🟩

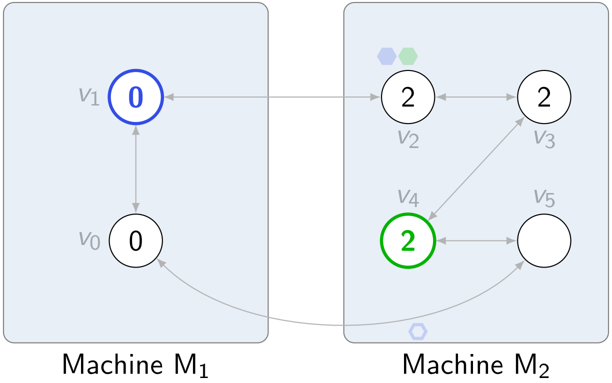
$M_2$ 🟩

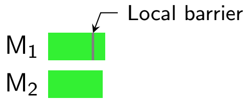
## BAP on WCC Algorithm



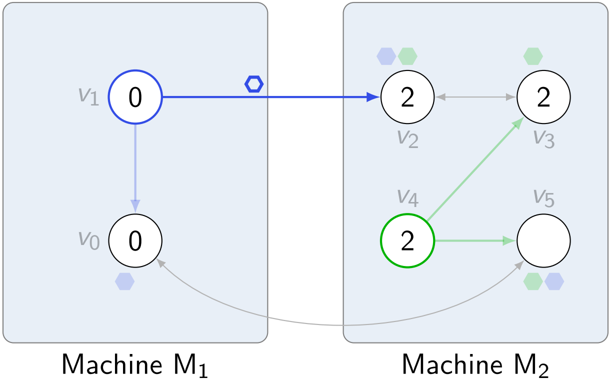


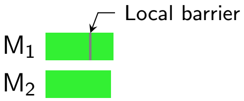
# BAP on WCC Algorithm



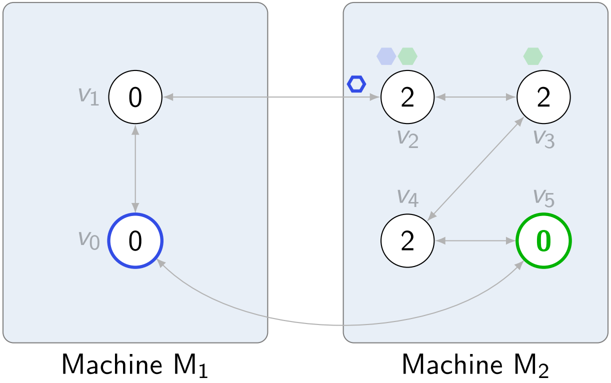


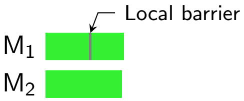
## BAP on WCC Algorithm



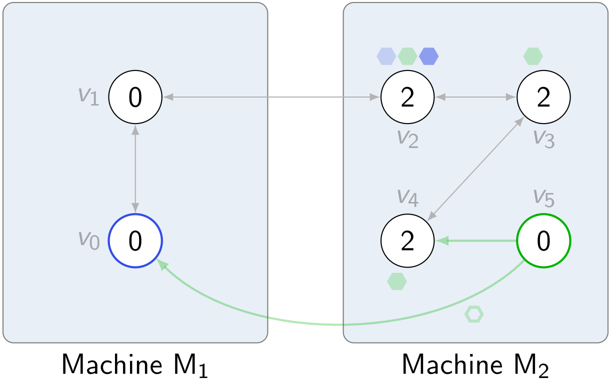


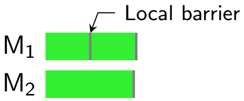
# BAP on WCC Algorithm



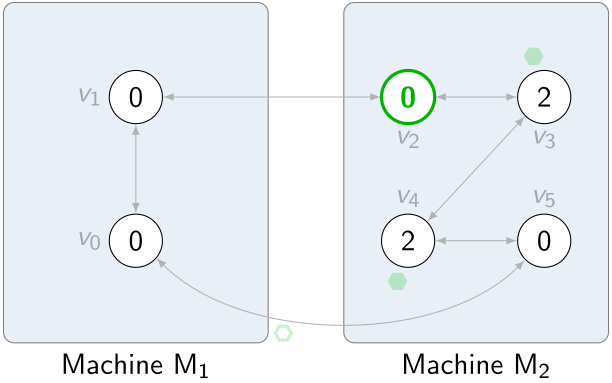


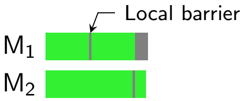
# BAP on WCC Algorithm



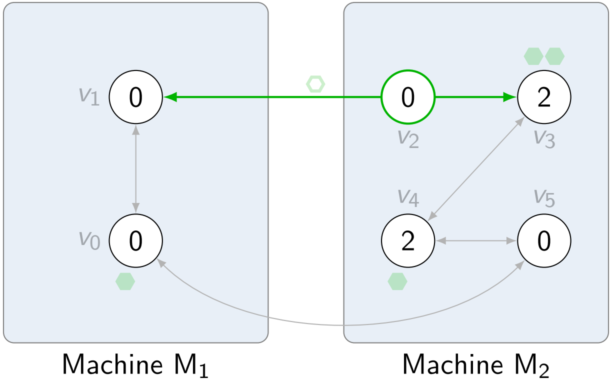


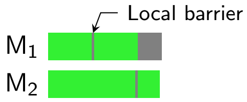
# BAP on WCC Algorithm



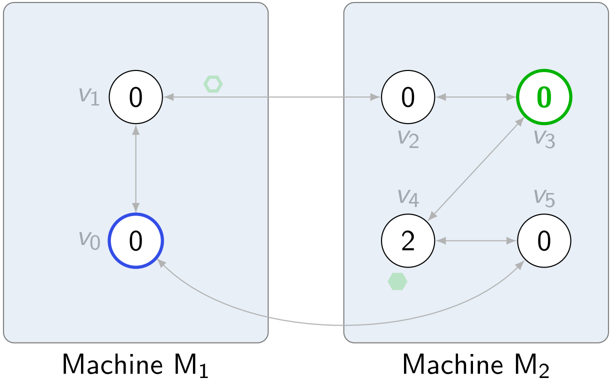


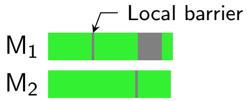
# BAP on WCC Algorithm



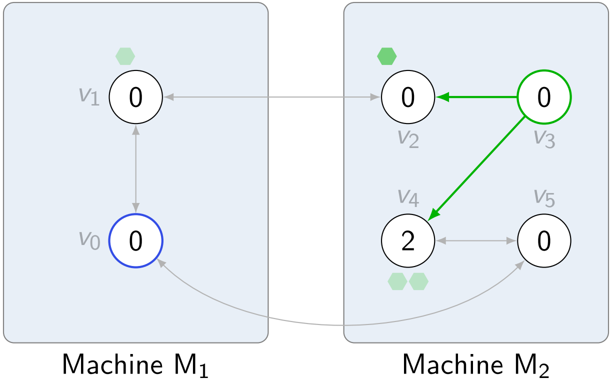


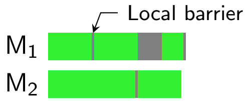
## BAP on WCC Algorithm



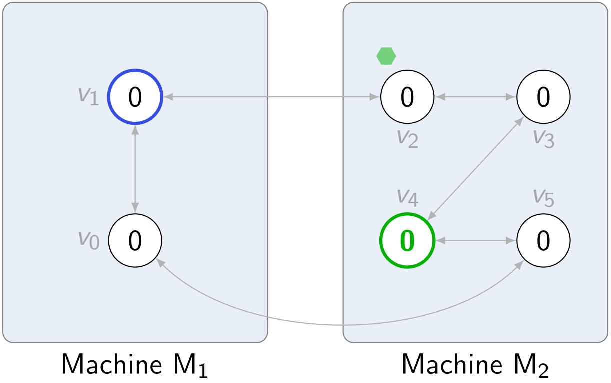


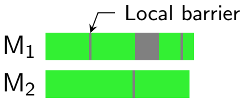
## BAP on WCC Algorithm



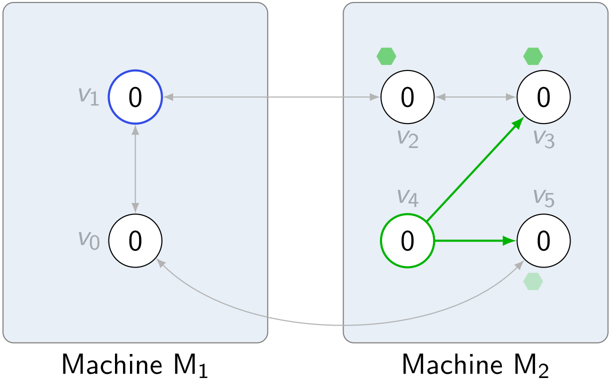


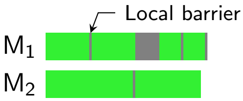
# BAP on WCC Algorithm



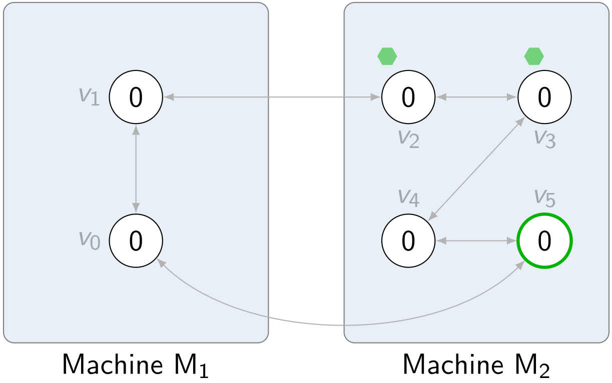


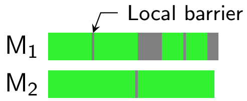
## BAP on WCC Algorithm



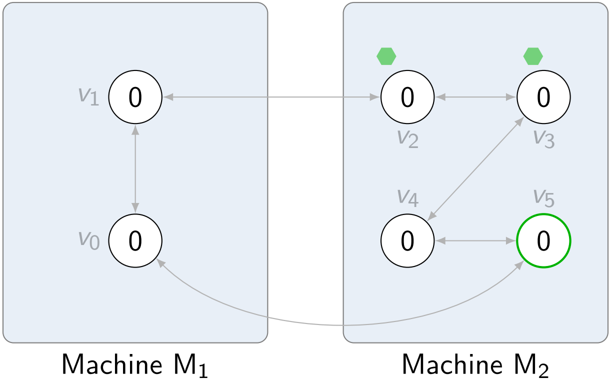


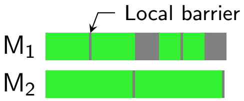
## BAP on WCC Algorithm



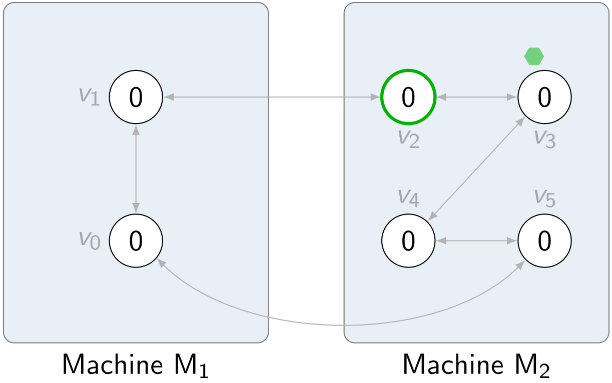


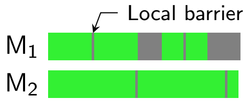
# BAP on WCC Algorithm



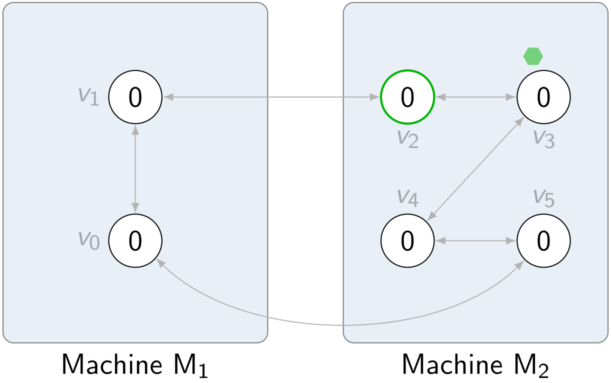


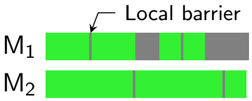
## BAP on WCC Algorithm



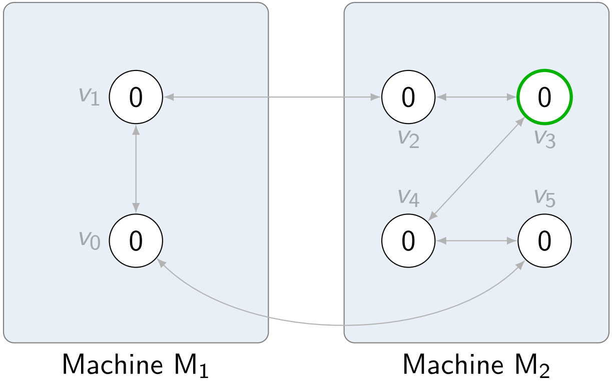


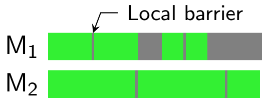
## BAP on WCC Algorithm



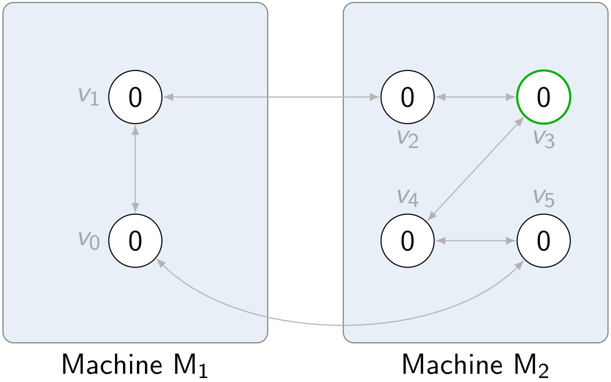


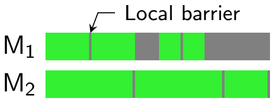
# BAP on WCC Algorithm



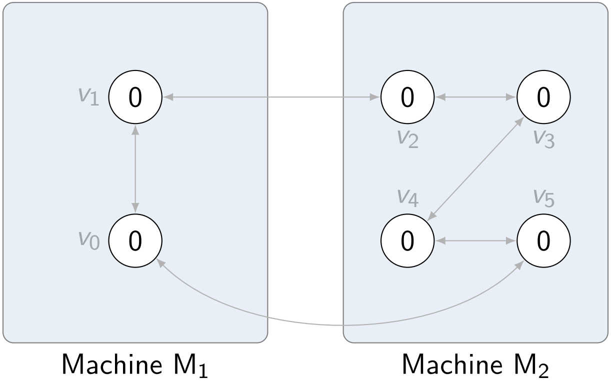


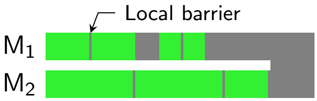
# BAP on WCC Algorithm



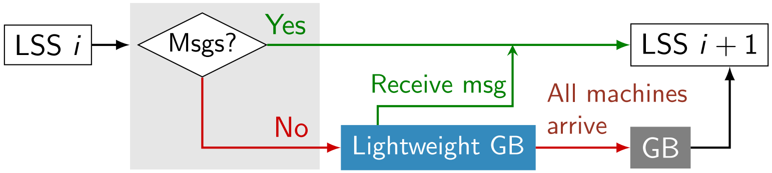


## BAP on WCC Algorithm





## Local Barriers



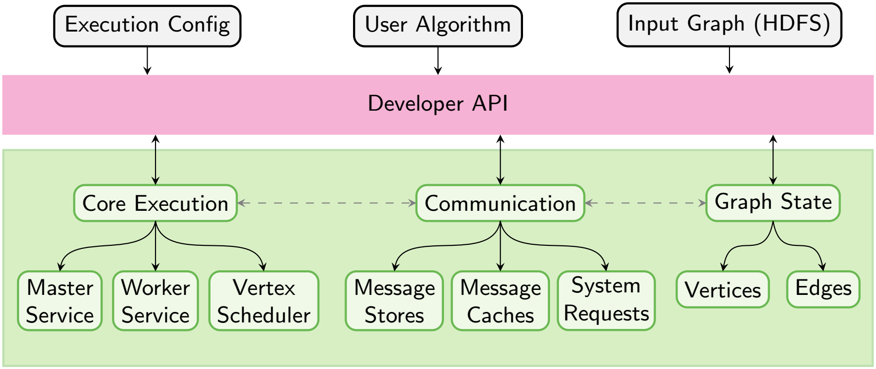
LSS: logical superstep.
GB: global barrier.

## Support for BSP Algorithms (O3)

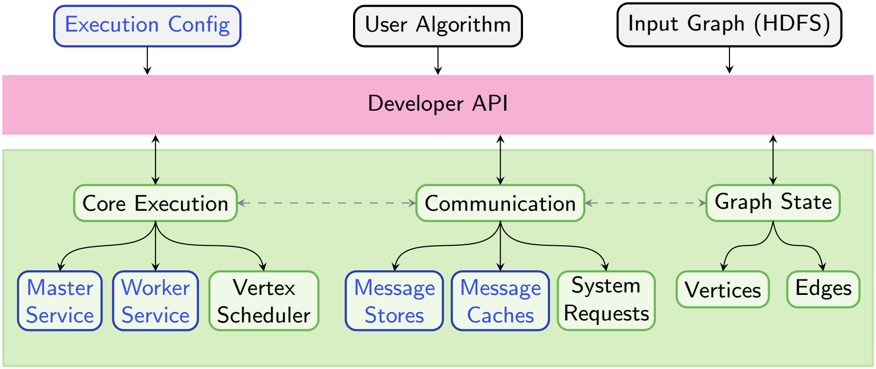
Retain support for BSP algorithms (**O3**):

- Use global barriers when necessary for correctness.
- Have full support for graph mutations.

## Giraph Architecture



## GiraphUC Architecture



GiraphUC implements the BAP model.

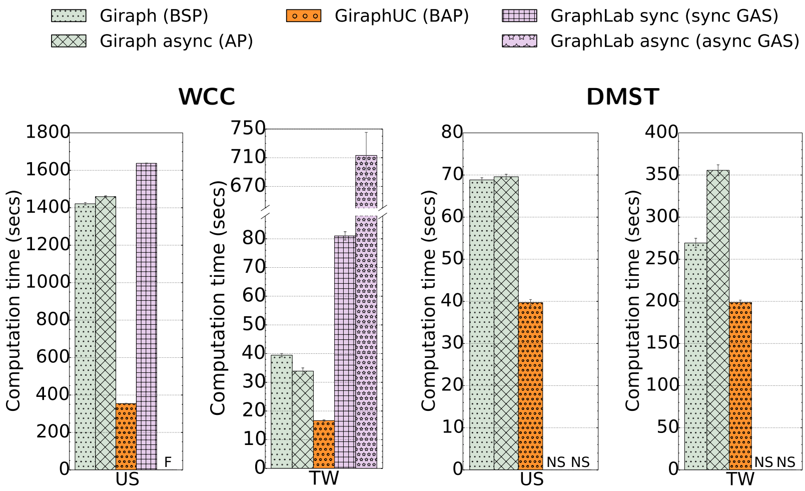
## Overview

## Setup

| | | | |
|---|---|---|---|
| | العربية | | |
| **US** | **AR** | **TW** | **UK** |
| | | $\lvert E \rvert = 1.46\text{B}$ | $\lvert E \rvert = 3.73\text{B}$ |

- 64 EC2 r3.xlarge instances (4 vCPUs, 30.5GB memory).
- Giraph 1.1.0 and GraphLab 2.2.
- Datasets loaded via random hash partitioning.

# Results



Giraph (BSP)
Giraph async (AP)
GiraphUC (BAP)
GraphLab sync (sync GAS)
GraphLab async (async GAS)
WCC
DMST
Computation time (secs)
US
TW
F
NS
NS
NS
NS

## Conclusion

| Computation Model | Reduce Stale Messages (O1) | Minimize Barriers (O2) | BSP Algorithm Support (O3) | Scalable |
|---|---|---|---|---|
| BSP/Sync GAS | ✗ | ✗ | ✓/✗ | ✓ |
| AP | ✓ | ✗ | ✗* | ✓ |
| Async GAS | ✓ | ✓ | ✗ | ✗ |
| **BAP** | ✓ | ✓ | ✓ | ✓ |

*We also enhanced the AP model to meet **O3**.

GiraphUC implements BAP and is:

- Up to **5**× faster than Giraph (BSP), Giraph async (AP), GraphLab sync (sync GAS).
- Up to **86**× faster than GraphLab async (async GAS).