Parallel Performance Optimizations on Unstructured Mesh-Based Simulations

Abhinav Sarje
asarje @ lbl.gov

02 June 2015
International Conference on Computational Science 2015
Reykjavík Iceland

Abhinav Sarje	Lawrence Berkeley National Laboratory
Sukhyun Song	University of Maryland College Park
Douglas Jacobsen	Los Alamos National Laboratory
Kevin Huck	University of Oregon
Jeffrey Hollingsworth	University of Maryland College Park
Allen Malony	University of Oregon
Samuel Willaims	Lawrence Berkeley National Laboratory
Leonid Oliker	Lawrence Berkeley National Laboratory

Introduction

- *Structured* and *unstructured* meshes in real-world simulations.
- Parallel applications – Not straightforward to partition unstructured meshes.
- Load balance directly related to mesh partitioning quality.
- Data exchange between processes through partition *ghost* or *halo* regions.
- Two key parallelization challenges:
  - Load imbalance across processes – mesh partitioning.
  - Unstructured data access patterns – data organization.

Introduction

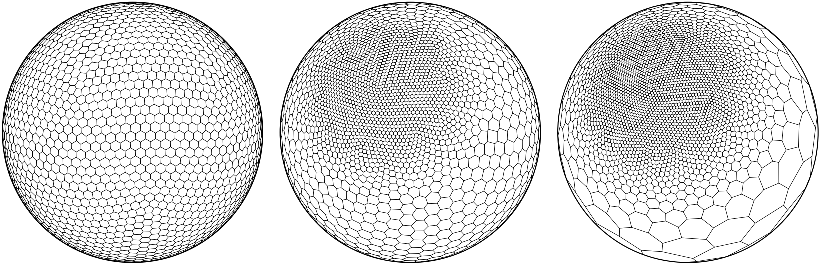
- *Structured* and *unstructured* meshes in real-world simulations.
- Parallel applications – Not straightforward to partition unstructured meshes.
- Load balance directly related to mesh partitioning quality.
- Data exchange between processes through partition *ghost* or *halo* regions.
- Two key parallelization challenges:
  - Load imbalance across processes – mesh partitioning.
  - Unstructured data access patterns – data organization.

Introduction

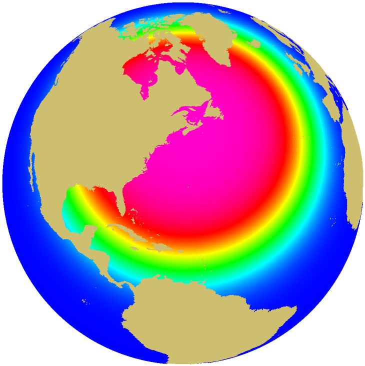
- *Structured* and *unstructured* meshes in real-world simulations.
- Parallel applications – Not straightforward to partition unstructured meshes.
- Load balance directly related to mesh partitioning quality.
- Data exchange between processes through partition *ghost* or *halo* regions.
- Two key parallelization challenges:
  - Load imbalance across processes – mesh partitioning.
  - Unstructured data access patterns – data organization.

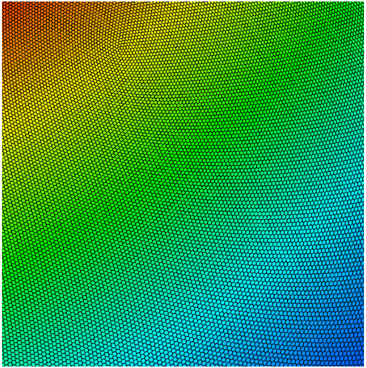
Ocean Modeling with MPAS-Ocean

- MPAS = **M**odel for **P**rediction **A**cross **S**cales. [Los Alamos]
- A multiscale method.
- Voronoi tessellation-based variable resolution mesh (SCVT).



Ocean Modeling with MPAS-Ocean





Ocean Modeling with MPAS-Ocean

- Major advantages of such unstructured mesh:
  - Offers variable resolutions. User defined density functions.
    - Focus on area of interest with high resolution.
    - Avoid unnecessary high-resolution computations in unwanted areas.
  - Smooth resolution transition regions.
  - Locally homogeneous/quasi-uniform coverage of spherical surfaces.
  - Preserve symmetry/isotropic nature of a spherical surface.
  - Naturally allows for discontinuities in the mesh.
  - Straightforward distortion-free mapping to 2D.
- A vertical quantization adds 3rd dimension, representing ocean depths.

Ocean Modeling with MPAS-Ocean

- Major advantages of such unstructured mesh:
  - Offers variable resolutions. User defined density functions.
    - Focus on area of interest with high resolution.
    - Avoid unnecessary high-resolution computations in unwanted areas.
  - Smooth resolution transition regions.
  - Locally homogeneous/quasi-uniform coverage of spherical surfaces.
  - Preserve symmetry/isotropic nature of a spherical surface.
  - Naturally allows for discontinuities in the mesh.
  - Straightforward distortion-free mapping to 2D.
- A vertical quantization adds 3rd dimension, representing ocean depths.

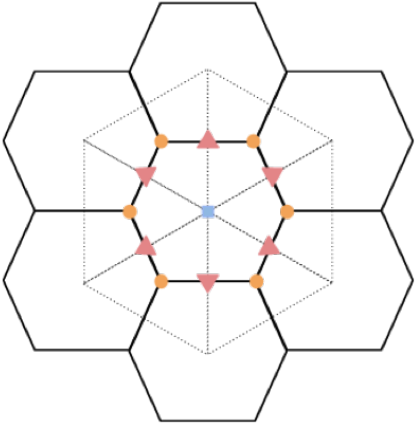
Ocean Modeling with MPAS-Ocean

- Major advantages of such unstructured mesh:
  - Offers variable resolutions. User defined density functions.
    - Focus on area of interest with high resolution.
    - Avoid unnecessary high-resolution computations in unwanted areas.
  - Smooth resolution transition regions.
  - Locally homogeneous/quasi-uniform coverage of spherical surfaces.
  - Preserve symmetry/isotropic nature of a spherical surface.
  - Naturally allows for discontinuities in the mesh.
  - Straightforward distortion-free mapping to 2D.
- A vertical quantization adds 3rd dimension, representing ocean depths.

Ocean Modeling with MPAS-Ocean

- Major advantages of such unstructured mesh:
  - Offers variable resolutions. User defined density functions.
    - Focus on area of interest with high resolution.
    - Avoid unnecessary high-resolution computations in unwanted areas.
  - Smooth resolution transition regions.
  - Locally homogeneous/quasi-uniform coverage of spherical surfaces.
  - Preserve symmetry/isotropic nature of a spherical surface.
  - Naturally allows for discontinuities in the mesh.
  - Straightforward distortion-free mapping to 2D.
- A vertical quantization adds 3rd dimension, representing ocean depths.

Ocean Modeling with MPAS-Ocean

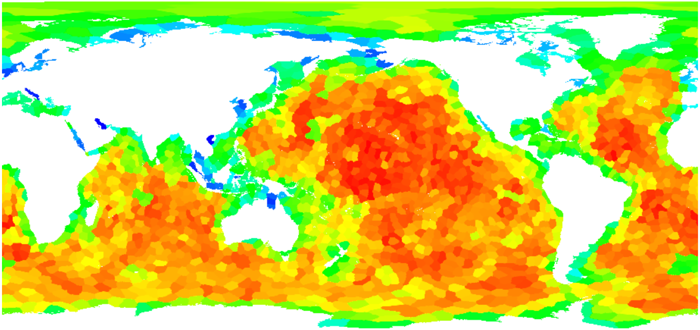


SCVT Cells and computed quantities.

Unstructured mesh partitioning ...

Mesh Partitioning and Load Imbalance

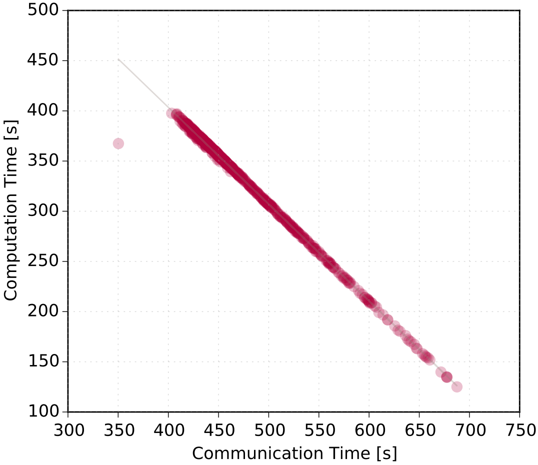
- Using a straight-forward graph partitioner, such as Metis.



Computational imbalance across partitions/processes

Mesh Partitioning and Load Imbalance

- High computation-communication imbalance across processes in a run with naive partitioning:



Mesh Partitioning and Load Imbalance

- Need for a better mesh partitioner.
- *Hypergraph* representations are known to model communication more accurately than graphs.
- Available partitioners generate a partitioning by,
  1. balancing the number of cells (or weights) across partitions, and
  2. minimizing the total number of edge cuts.
- Problem:
  - Cost due to halo cells is not considered.
  - Unstructured nature makes halo region costs highly variable across partitions.
  - *Deep* halo regions magnify the effects, making them an important factor for load balancing.

Mesh Partitioning and Load Imbalance

- Need for a better mesh partitioner.
- *Hypergraph* representations are known to model communication more accurately than graphs.
- Available partitioners generate a partitioning by,
  1. balancing the number of cells (or weights) across partitions, and
  2. minimizing the total number of edge cuts.
- Problem:
  - Cost due to halo cells is not considered.
  - Unstructured nature makes halo region costs highly variable across partitions.
  - *Deep* halo regions magnify the effects, making them an important factor for load balancing.

Mesh Partitioning and Load Imbalance

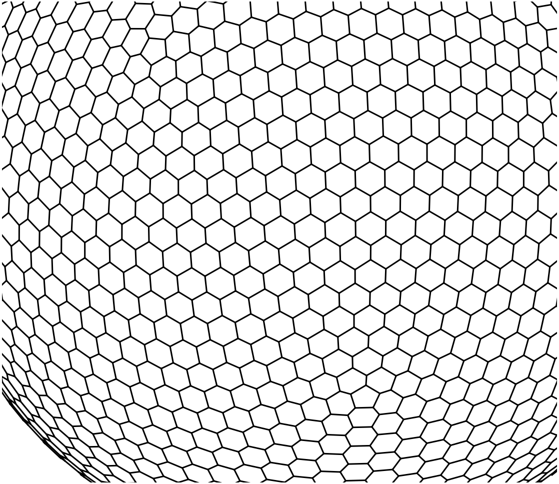
- Need for a better mesh partitioner.
- *Hypergraph* representations are known to model communication more accurately than graphs.
- Available partitioners generate a partitioning by,
  1. balancing the number of cells (or weights) across partitions, and
  2. minimizing the total number of edge cuts.
- Problem:
  - Cost due to halo cells is not considered.
  - Unstructured nature makes halo region costs highly variable across partitions.
  - *Deep* halo regions magnify the effects, making them an important factor for load balancing.

Mesh Partitioning and Load Imbalance

- Need for a better mesh partitioner.
- *Hypergraph* representations are known to model communication more accurately than graphs.
- Available partitioners generate a partitioning by,
  1. balancing the number of cells (or weights) across partitions, and
  2. minimizing the total number of edge cuts.
- Problem:
  - Cost due to halo cells is not considered.
  - Unstructured nature makes halo region costs highly variable across partitions.
  - *Deep* halo regions magnify the effects, making them an important factor for load balancing.

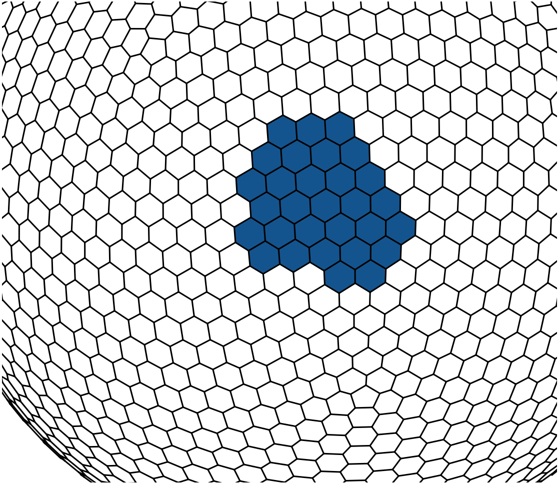
Mesh Partitioning and Load Imbalance

- Input mesh ...



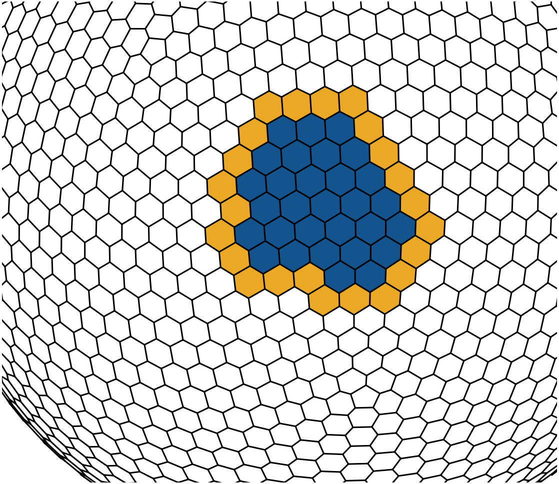
Mesh Partitioning and Load Imbalance

- A partition ...



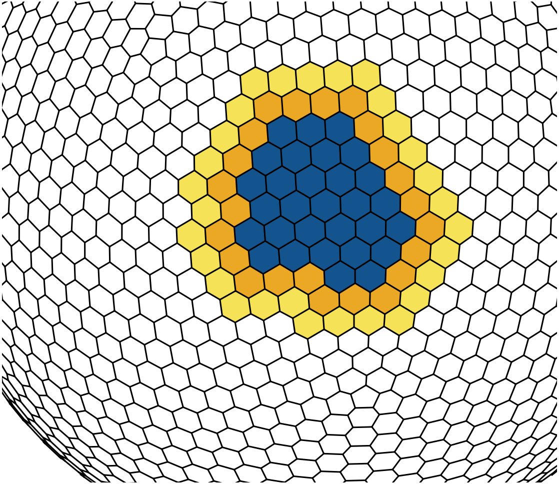
Mesh Partitioning and Load Imbalance

- 1-Halo region ...



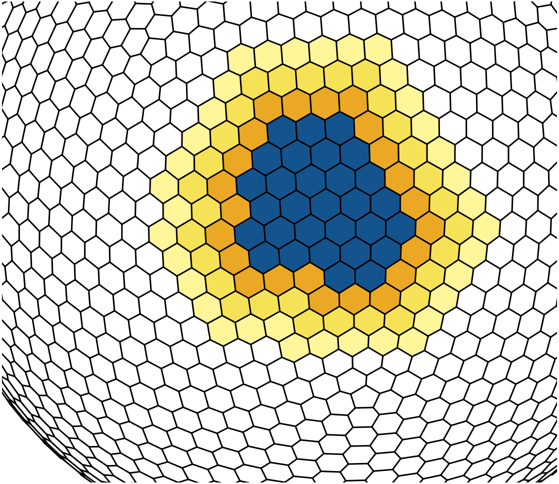
Mesh Partitioning and Load Imbalance

- 2-Halo region ...



Mesh Partitioning and Load Imbalance

- 3-Halo region ...



Partitioning-Based Cost Modeling

In a partitioning, for a partition k,

$$\text{computation cost, } C_\alpha = \frac{1}{F(p)}\left(\sum_{i}^{i\in N_k} w_i + a\sum_{i}^{i\in H_k} w_i\right)$$

$$\text{communication cost, } C_\beta = \frac{1}{F(p)}\frac{h_k}{b_k} + \left(\max_{i\in[1,p]}(c_i) - c_k\right)$$

Partitioning-Based Cost Modeling

In a partitioning, for a partition k,

$$\text{computation cost, } C_\alpha = \frac{1}{F(p)} \left(\sum_{i}^{i \in N_k} w_i + a \sum_{i}^{i \in H_k} w_i \right)$$

$$\text{communication cost, } C_\beta = \frac{1}{F(p)} \frac{h_k}{b_k} + \left(\max_{i \in [1,p]} (c_i) - c_k \right)$$

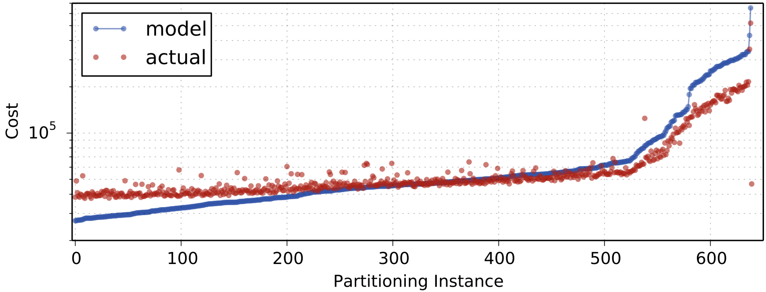
Partitioning-Based Cost Modeling

In a partitioning, for a partition k,

$$\text{computation cost, } C_\alpha = \frac{1}{F(p)} \left(\sum_{i}^{i \in N_k} w_i + a \sum_{i}^{i \in H_k} w_i \right)$$

$$\text{communication cost, } C_\beta = \frac{1}{F(p)} \frac{h_k}{b_k} + \left(\max_{i \in [1,p]} (c_i) - c_k \right)$$

Partitioning-Based Cost Modeling



A Halo-Aware Partitioning Approach

```
procedure HaloAwarePartition(𝒢)
    Construct sparse matrix A representing 𝒢
    Compute A^2, ··· A^l, and A_{1...l} = Σ^l A^i
    Construct hypergraph ℋ_0 for A_{1...l}
    while not converged do
        Compute partitioning P_i of ℋ_i; construct halos for each partition in P_i
        Compute cost prediction for each partition k
        Assign weights to the cells, distributing halo cost equally among partition cells
        Compute total partition weights W_k
        Compute imbalance measure, f_i = (1 − min_k(W_k) / max_k(W_k))
        Accept P_i with probability m = min(1, e^((f_{i−1} − f_i)/2))
        if P_i is accepted then
            Update ℋ_i with the new cell weights to construct ℋ_(i+1)
        else
            Reject P_i by setting P_i = P_{i−1} and f_i = f_{i−1}
        end if
    end while
    Output last accepted partitioning as the result
end procedure
```

A Halo-Aware Partitioning Approach

procedure HALOAWAREPARTITION($\mathcal{G}$)
  Construct sparse matrix A representing $\mathcal{G}$
  Compute $A^2, \cdots A^l$, and $A_{1\ldots l} = \sum^l A^i$
  Construct hypergraph $\mathcal{H}_0$ for $A_{1\ldots l}$
  while not converged **do**
    Compute partitioning P_i of $\mathcal{H}_i$; construct halos for each partition in P_i
    Compute cost prediction for each partition k
    Assign weights to the cells, distributing halo cost equally among partition cells
    Compute total partition weights W_k
    Compute imbalance measure, $f_i = \left(1 - \frac{\min_k(W_k)}{\max_k(W_k)}\right)$
    Accept P_i with probability $m = \min\left(1, e^{\left(f_{i-1} - f_i\right)/2}\right)$
    if P_i is accepted **then**
      Update $\mathcal{H}_i$ with the new cell weights to construct $\mathcal{H}_{(i+1)}$
    else
      Reject P_i by setting $P_i = P_{i-1}$ and $f_i = f_{i-1}$
    end if
  end while
  Output last accepted partitioning as the result
end procedure

A Halo-Aware Partitioning Approach

procedure HALOAWAREPARTITION($\mathcal{G}$)
  Construct sparse matrix A representing $\mathcal{G}$
  Compute $A^2, \cdots A^l$, and $A_{1\ldots l} = \sum^l A^i$
  Construct hypergraph $\mathcal{H}_0$ for $A_{1\ldots l}$
  while not converged **do**
    Compute partitioning P_i of $\mathcal{H}_i$; construct halos for each partition in P_i
    Compute cost prediction for each partition k
    Assign weights to the cells, distributing halo cost equally among partition cells
    Compute total partition weights W_k
    Compute imbalance measure, $f_i = \left(1 - \frac{\min_k(W_k)}{\max_k(W_k)}\right)$
    Accept P_i with probability $m = \min\left(1, e^{\left(f_{i-1} - f_i\right)/2}\right)$
    if P_i is accepted **then**
      Update $\mathcal{H}_i$ with the new cell weights to construct $\mathcal{H}_{(i+1)}$
    else
      Reject P_i by setting $P_i = P_{i-1}$ and $f_i = f_{i-1}$
    end if
  end while
  Output last accepted partitioning as the result
end procedure

A Halo-Aware Partitioning Approach

procedure HALOAWAREPARTITION($\mathcal{G}$)
    Construct sparse matrix A representing $\mathcal{G}$
    Compute $A^2, \cdots A^l$, and $A_{1\ldots l} = \sum^l A^i$
    Construct hypergraph $\mathcal{H}_0$ for $A_{1\ldots l}$
    while not converged **do**
        Compute partitioning P_i of $\mathcal{H}_i$; construct halos for each partition in P_i
        Compute cost prediction for each partition k
        Assign weights to the cells, distributing halo cost equally among partition cells
        Compute total partition weights W_k
        Compute imbalance measure, $f_i = \left(1 - \frac{\min_k(W_k)}{\max_k(W_k)}\right)$
        Accept P_i with probability $m = \min\left(1, e^{\left(f_{i-1} - f_i\right)/2}\right)$
        if P_i is accepted **then**
            Update $\mathcal{H}_i$ with the new cell weights to construct $\mathcal{H}_{(i+1)}$
        else
            Reject P_i by setting $P_i = P_{i-1}$ and $f_i = f_{i-1}$
        end if
    end while
    Output last accepted partitioning as the result
end procedure

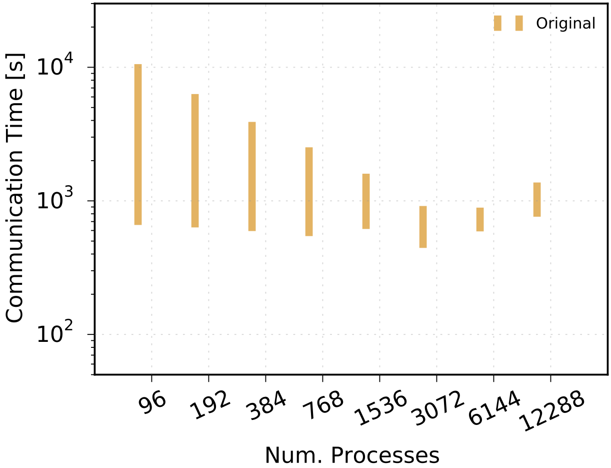
A Halo-Aware Partitioning Approach

procedure HALOAWAREPARTITION($\mathcal{G}$)
  Construct sparse matrix A representing $\mathcal{G}$
  Compute $A^2, \cdots A^l$, and $A_{1\ldots l} = \sum^l A^i$
  Construct hypergraph $\mathcal{H}_0$ for $A_{1\ldots l}$
  while not converged **do**
    Compute partitioning P_i of $\mathcal{H}_i$; construct halos for each partition in P_i
    Compute cost prediction for each partition k
    Assign weights to the cells, distributing halo cost equally among partition cells
    Compute total partition weights W_k
    Compute imbalance measure, $f_i = \left(1 - \frac{\min_k(W_k)}{\max_k(W_k)}\right)$
    Accept P_i with probability $m = \min\left(1, e^{\left(f_{i-1} - f_i\right)/2}\right)$
    if P_i is accepted **then**
      Update $\mathcal{H}_i$ with the new cell weights to construct $\mathcal{H}_{(i+1)}$
    else
      Reject P_i by setting $P_i = P_{i-1}$ and $f_i = f_{i-1}$
    end if
  end while
  Output last accepted partitioning as the result
end procedure

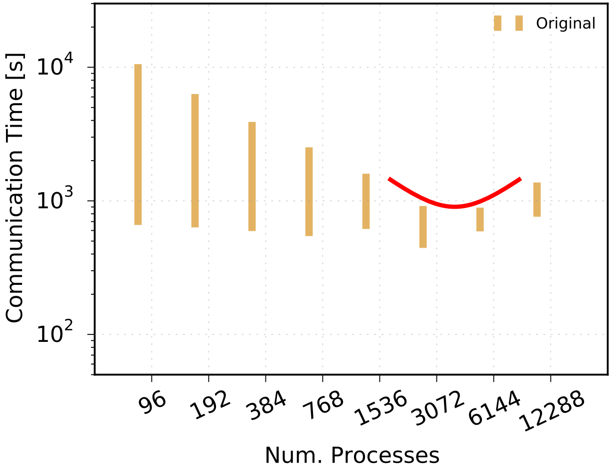
A Halo-Aware Partitioning Approach

procedure HALOAWAREPARTITION($\mathcal{G}$)
    Construct sparse matrix A representing $\mathcal{G}$
    Compute $A^2, \cdots A^l$, and $A_{1\ldots l} = \sum^l A^i$
    Construct hypergraph $\mathcal{H}_0$ for $A_{1\ldots l}$
    while not converged **do**
        Compute partitioning P_i of $\mathcal{H}_i$; construct halos for each partition in P_i
        Compute cost prediction for each partition k
        Assign weights to the cells, distributing halo cost equally among partition cells
        Compute total partition weights W_k
        Compute imbalance measure, $f_i = \left(1 - \frac{\min_k(W_k)}{\max_k(W_k)}\right)$
        Accept P_i with probability $m = \min\left(1, e^{\left(f_{i-1} - f_i\right)/2}\right)$
        if P_i is accepted **then**
            Update $\mathcal{H}_i$ with the new cell weights to construct $\mathcal{H}_{(i+1)}$
        else
            Reject P_i by setting $P_i = P_{i-1}$ and $f_i = f_{i-1}$
        end if
    end while
    Output last accepted partitioning as the result
end procedure

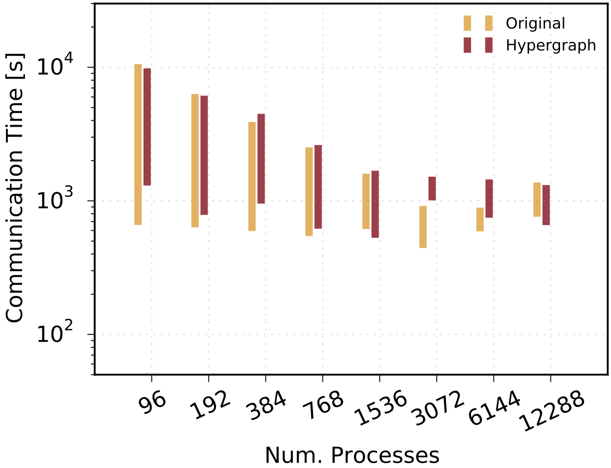
Performance Improvements with Better Partitioning



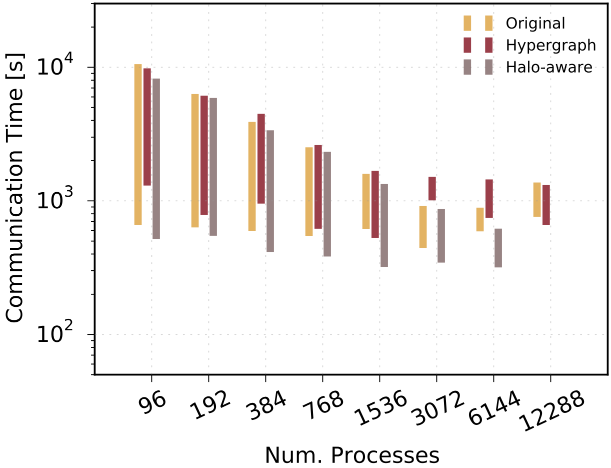
Performance Improvements with Better Partitioning



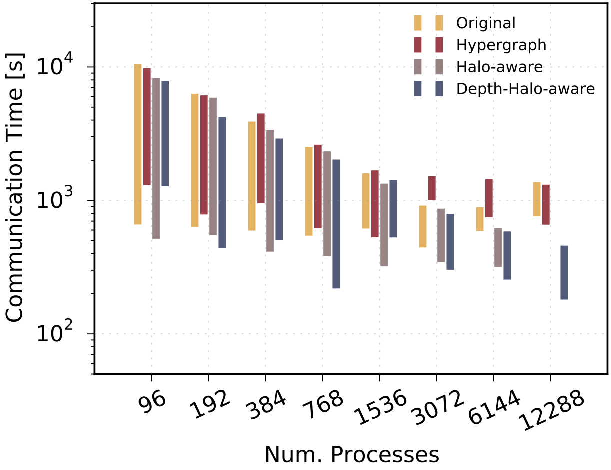
Performance Improvements with Better Partitioning



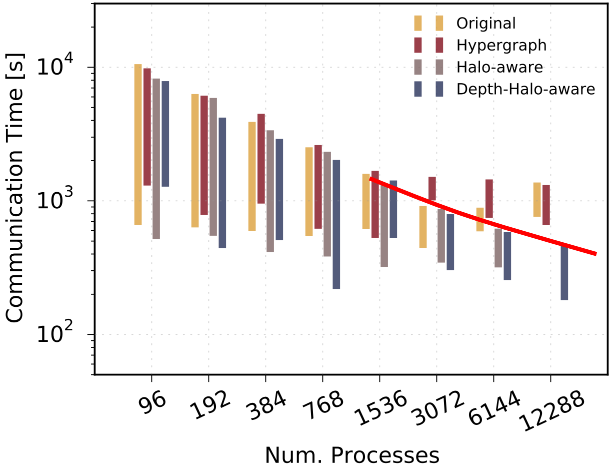
Performance Improvements with Better Partitioning



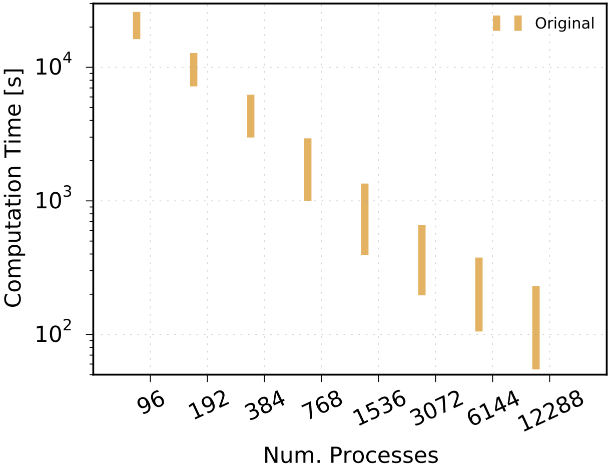
Performance Improvements with Better Partitioning



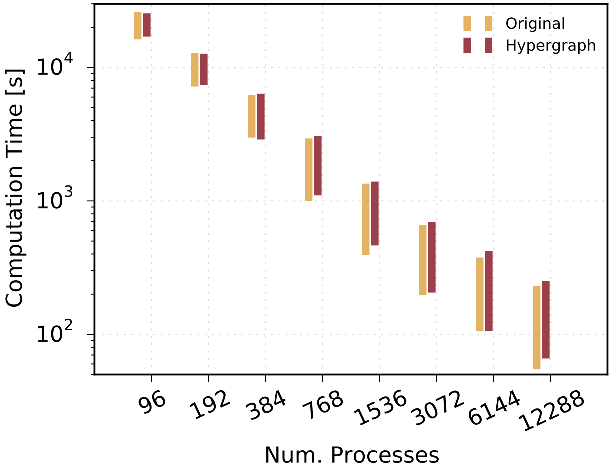
Performance Improvements with Better Partitioning



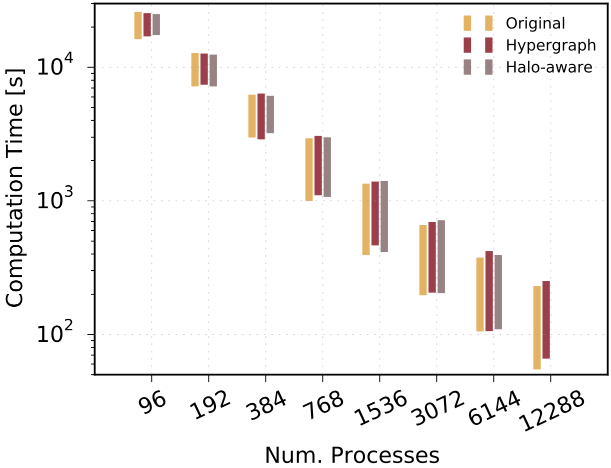
Performance Improvements with Better Partitioning



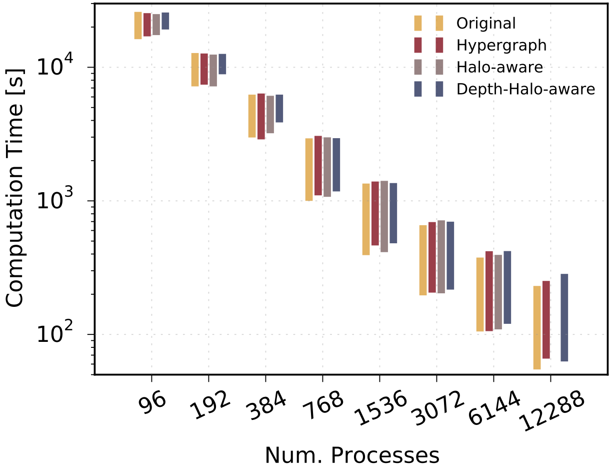
Performance Improvements with Better Partitioning



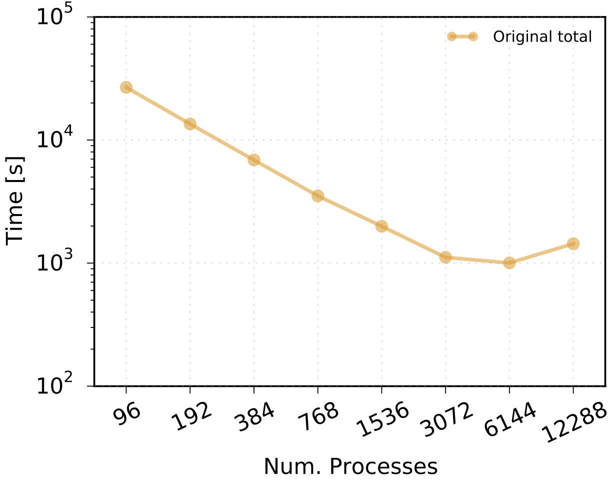
Performance Improvements with Better Partitioning



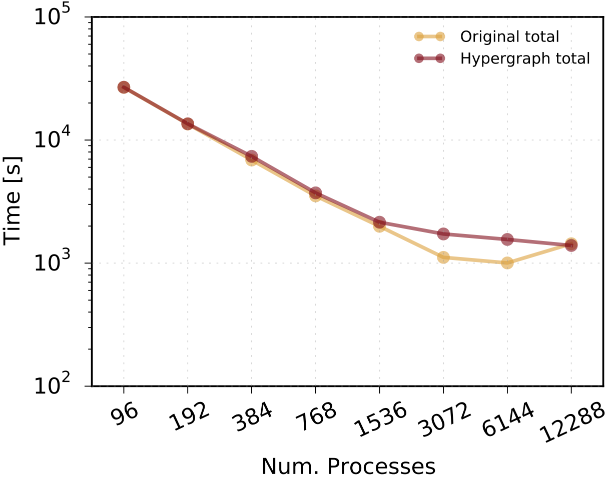
Performance Improvements with Better Partitioning



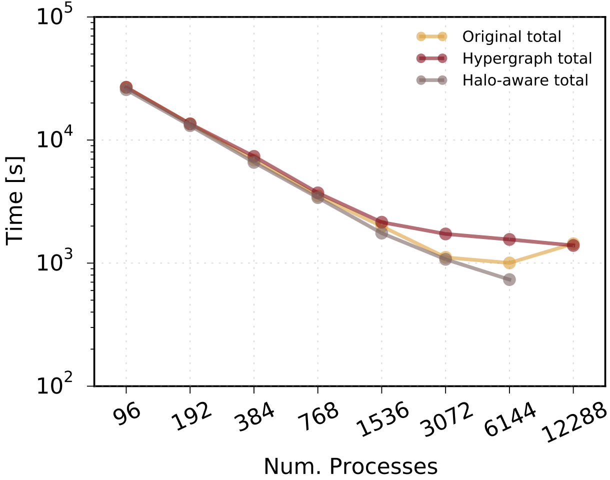
Performance Improvements with Better Partitioning



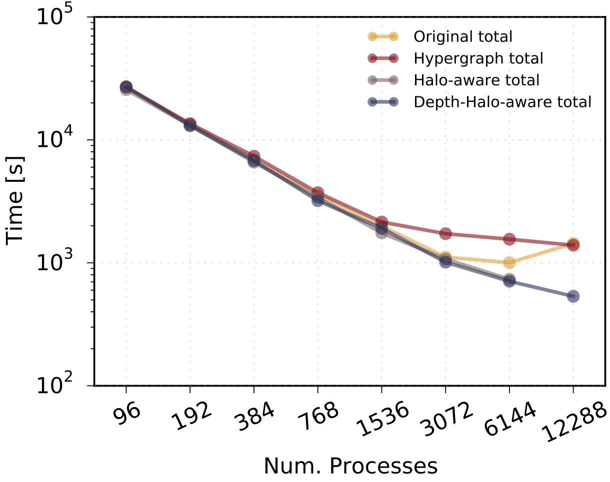
Performance Improvements with Better Partitioning



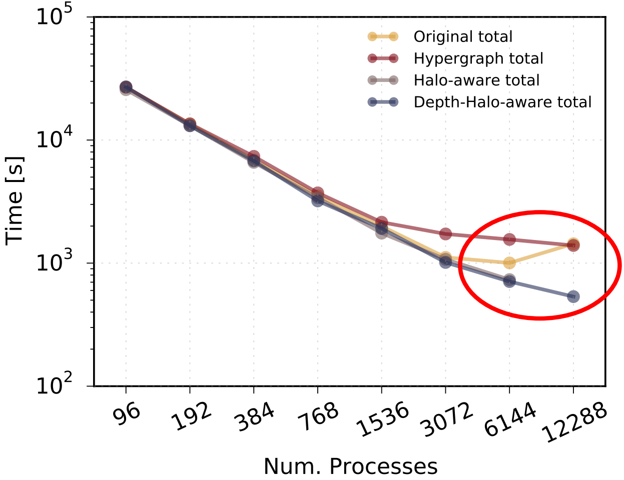
Performance Improvements with Better Partitioning



Performance Improvements with Better Partitioning

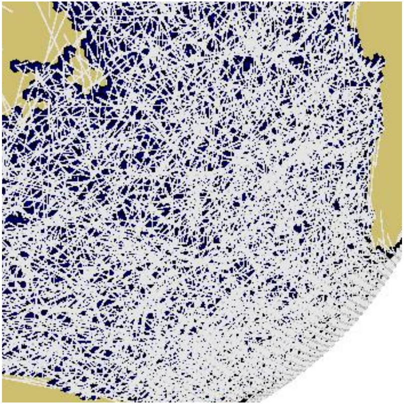


Performance Improvements with Better Partitioning

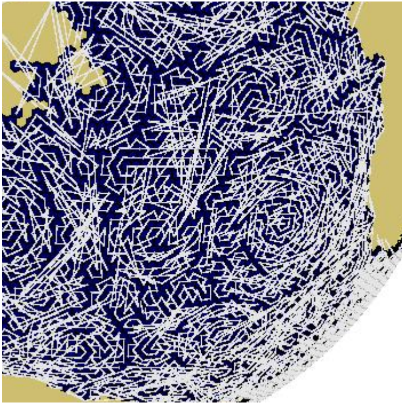


Unstructured data organization ...

Data Locality with Data Ordering

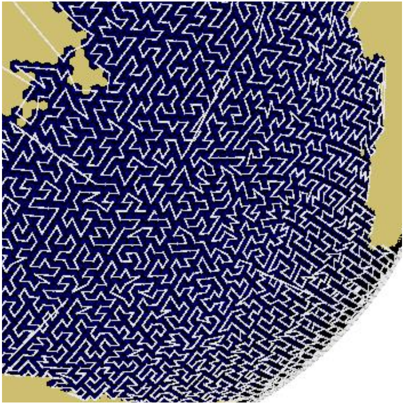


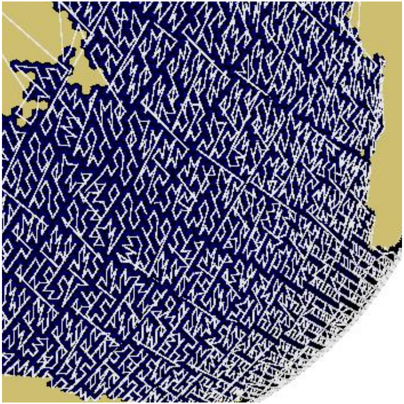
A complete random organization



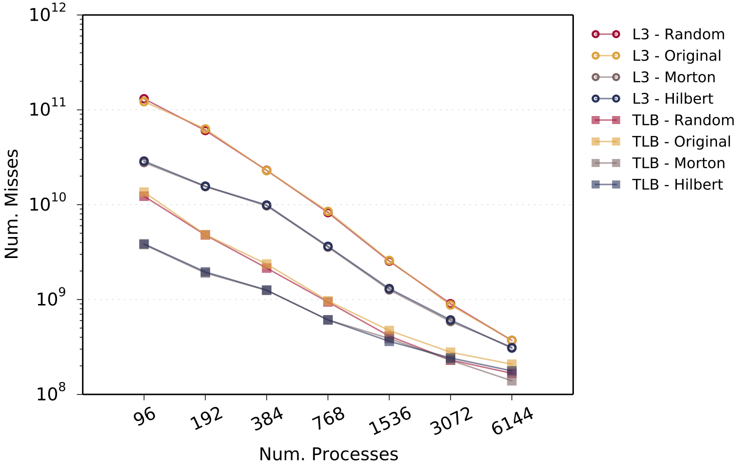
Original/Reverse Cuthill-McKee

Ordering Unstructured Data with Space Filling Curves

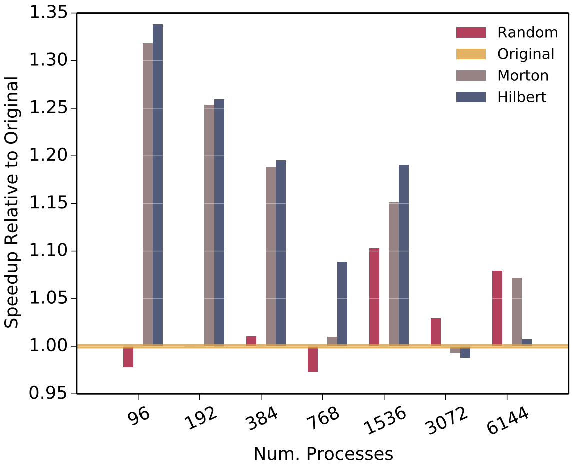

Hilbert SFC


Morton/Z-SFC

Performance Improvements with Data Re-ordering: Cache Usage



Performance Improvements with Data Re-ordering



End Notes

- Overall improved performance by up to 2.2$\times$.
- Improved scaling.
- Enable increased resolution and throughput of high resolution meshes.
- Achieve high SYPD (Simulated Year Per Day.)
- Enable higher accuracy with high resolution.

- Partitioning and ordering methods are generic to apply to other unstructured meshes.
- Collaborations ... ?

Acknowledgements

- Thanks to Aydin Buluc and Umit Catalyurek for discussions on mesh partitioning.
- Authors from LBNL were supported by DOE ASCR under contract number DE-AC02-05CH11231.
- Authors from University of Oregon were supported by DOE SciDAC grant DE-SC0006723.
- D. Jacobsen was supported by DOE Office of BER.
- Resources at NERSC were used, supported by DOE Office of Science under Contract No. DE-AC02-05CH11231.

Thank you!