情報処理学会グラフィクスとCAD研究報告, 2007-CG-128, Vol.2007, No.84, pp.65-70 (2007)

グリッドコンピューティングを用いた大規模ボリュームデータの可視化手法に関する研究

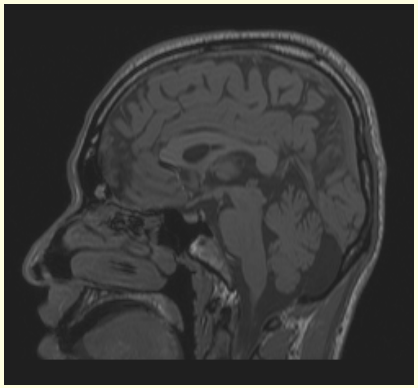
広島大学

檜垣 徹, 西橋 邦彦, 岡部 憲史,
玉木 徹, 金田 和文

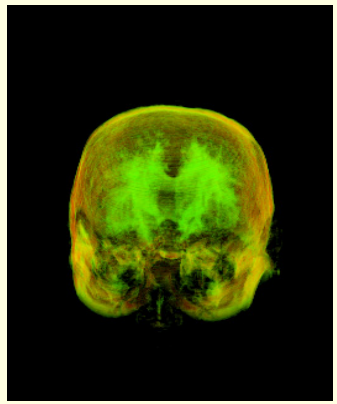
ボリュームレンダリング

- 3次元表示により直観的に、内部まで観察が可能

例）医療応用



頭部断面画像



ボリュームレンダリング
による表示

- ボリュームレンダリングの問題点
  - 計算コストが高い
  - ボリュームデータの一辺の解像度の3乗に比例

並列コンピューティング

- クラスタ
  - 導入時のコストが高い
  - 計算能力は導入した時点で決定される
- グリッドコンピューティング
  - 既存の計算機を使用するため導入コストは低い
  - グリッド網を広げることで計算能力を拡張可能
  - 必要なとき必要なだけ利用可能

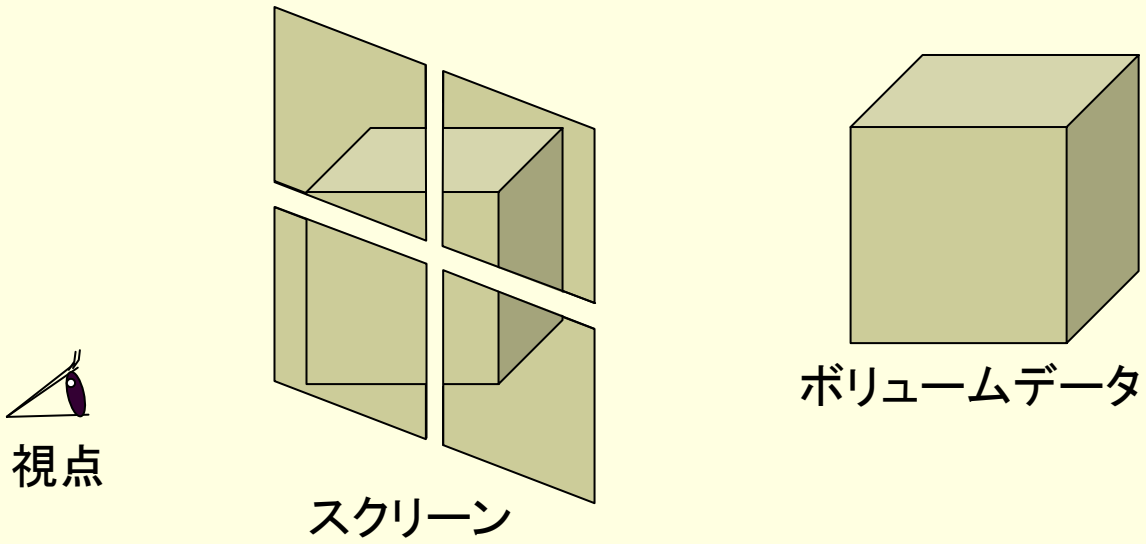
目的

グリッドコンピューティングを用いた
ボリュームレンダリングの高速化

将来性の高い
グリッドコンピューティングを用いた
ボリュームレンダリング手法を開発

ボリュームレンダリングの分散化

- スクリーンを分割
  - 任意視点からのレンダリングを行うには
    計算ノードごとにボリュームデータ全体が必要
  - 大規模データを取り扱うには大きなメモリが必要

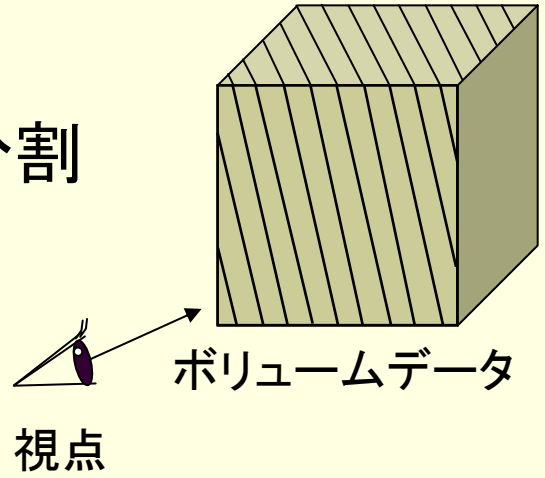


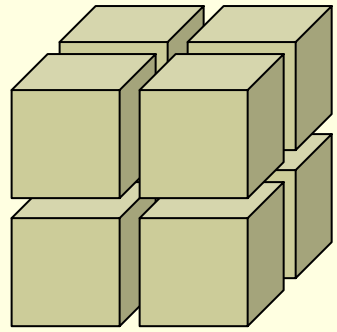
ボリュームレンダリングの分散化

- ボリュームデータを分割
  - スクリーンに平行なスライスで分割
    - 視点に依存する
    - 可視性を把握しやすい
  - 賽の目に分割
    - 視点に依存しない
    - 可視性を把握しにくい

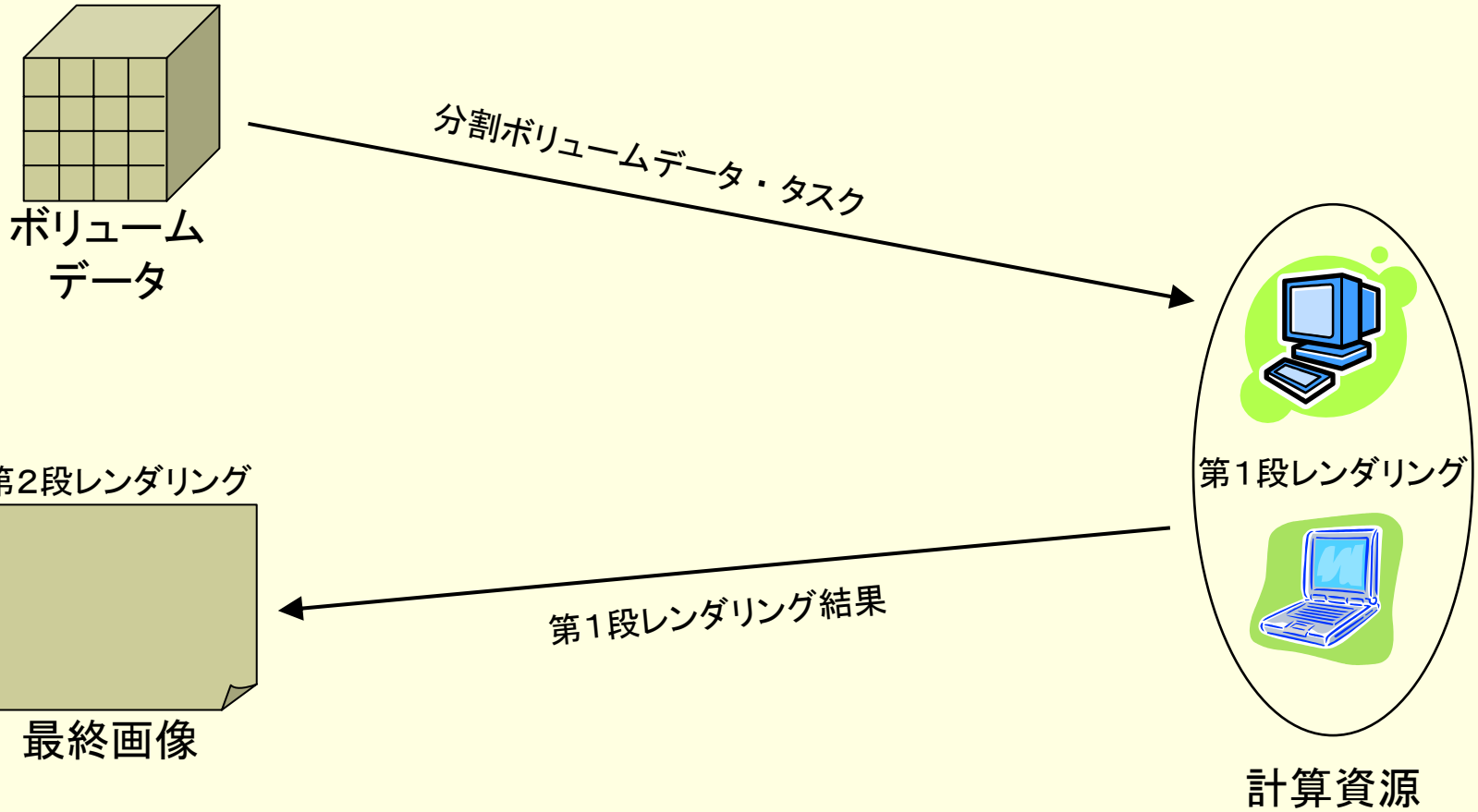
村木(2002)
クラスタで並列レンダリング
専用ハードウェアを用いて合成

松井(2004)
クラスタで並列レンダリング
スクリーンからの距離順に合成





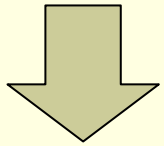
基本的な分散処理



第２段レンダリングには可視性に基づく順序がある
第２段レンダリングの順序に従いタスクを投入

グリッドコンピューティングの特徴

- 導入コストが低い
  → 専用ハードウェアの導入は好ましくない
- 計算資源が不均一
- 提供される資源が時間とともに変化する
  → タスクが投入した順に帰ってこない可能性
  静的に決定した順番に従うと効率低下



動的な可視性・投入タスクの管理が必要

提案手法

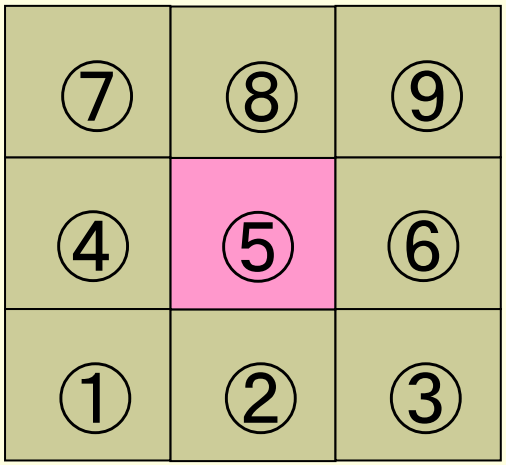
- 分割したボリュームデータの可視性を
  動的に把握・管理する機構

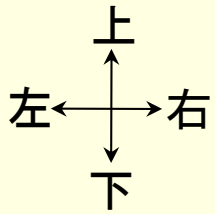
オブスタクルフラグ

- 遮蔽するもの（＝未レンダリングのボリュームデータ）
  が存在するか否か をフラグとして保存
- 可視性が変化するたびにフラグを更新し
  動的に最新の可視性を把握
- オブスタクルフラグから可視性を確認し
  可視性が最も高いものをタスクとして投入

オブスタクルフラグ

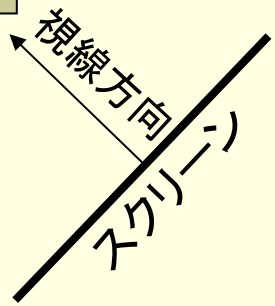
- 分割ボリュームデータの面の数のビット数（=6bit）必要
- 各分割ボリュームデータに1つずつ
- 全てのフラグが0 = 遮蔽物が無い = 第2段レンダリング可能





※説明の都合上、2次元で示す

①～⑨はそれぞれ
分割ボリュームデータ



⑤のオブスタクルフラグ	
上	0
下	1
左	0
右	1

対応する面（辺）に遮蔽物が…

存在するならばフラグは 1
存在しなければフラグは 0

オブスタクルフラグの使用例

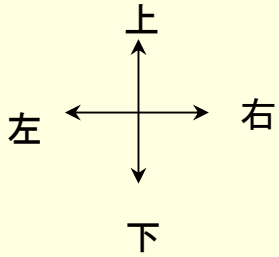
⑦	⑧	⑨
④	⑤	⑥
①	②	③

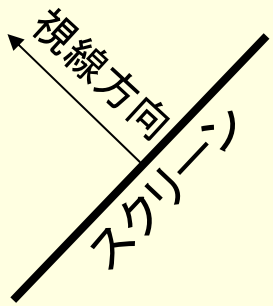
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	1	1	1	1
左	0	0	0	0	0	0	0	0	0
右	1	1	0	1	1	0	1	1	0

: 第２段レンダリング済

: 第１段レンダリング中

: 未処理





オブスタクルフラグの使用例

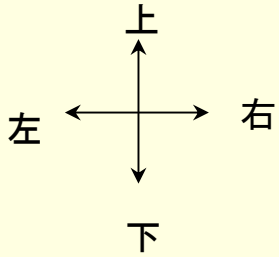
⑦	⑧	⑨
④	⑤	⑥
①	②	③

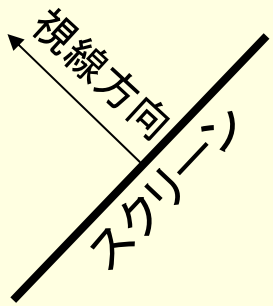
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	1	1	1	1
左	0	0	0	0	0	0	0	0	0
右	1	1	0	1	1	0	1	1	0

: 第２段レンダリング済

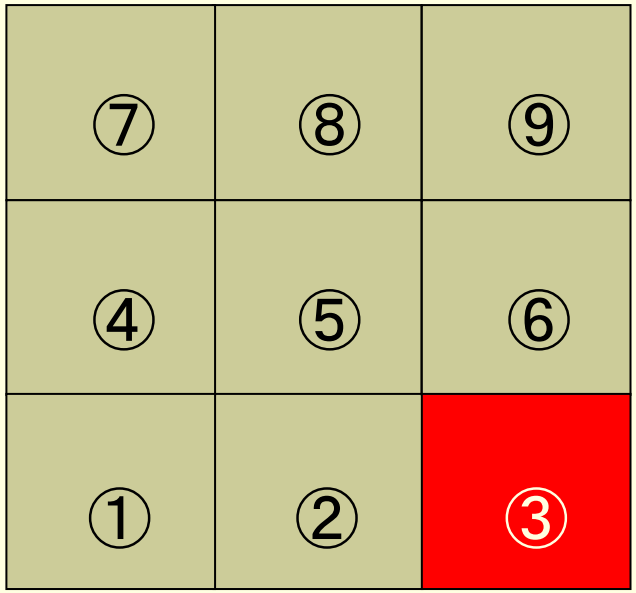
: 第１段レンダリング中

: 未処理





オブスタクルフラグの使用例

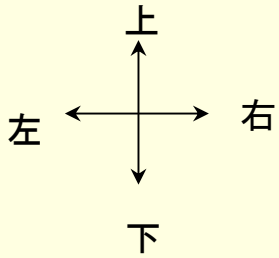


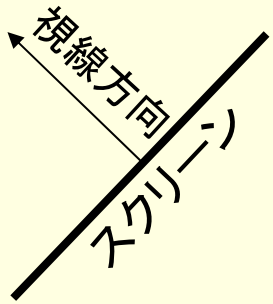
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	0	1	1	1
左	0	0	0	0	0	0	0	0	0
右	1	0	0	1	1	0	1	1	0

: 第２段レンダリング済

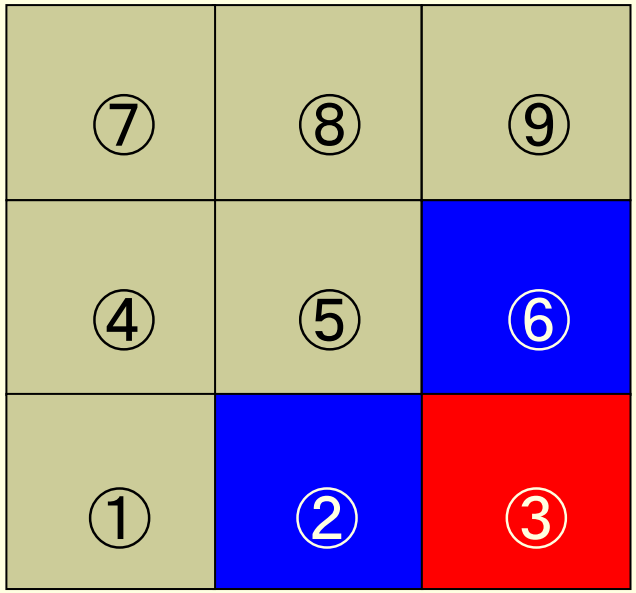
: 第１段レンダリング中

: 未処理





オブスタクルフラグの使用例

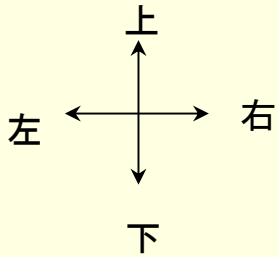


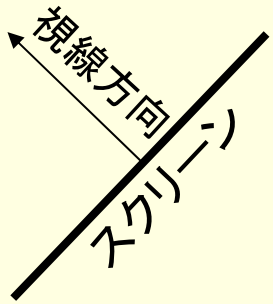
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	0	1	1	1
左	0	0	0	0	0	0	0	0	0
右	1	0	0	1	1	0	1	1	0

: 第２段レンダリング済

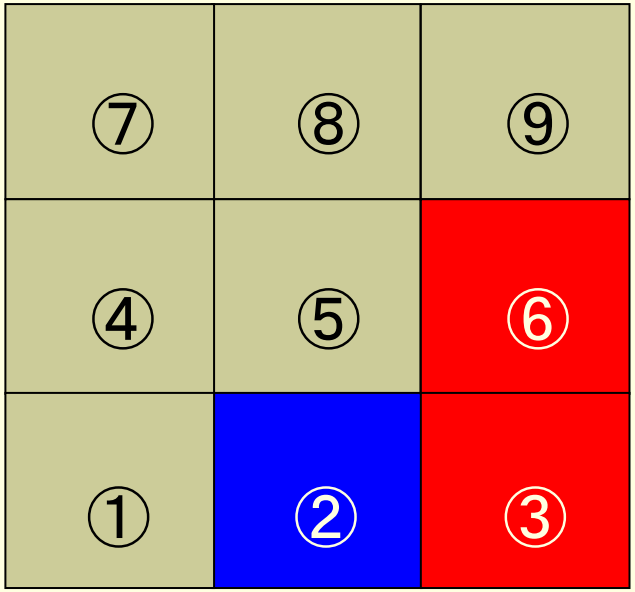
: 第１段レンダリング中

: 未処理





オブスタクルフラグの使用例

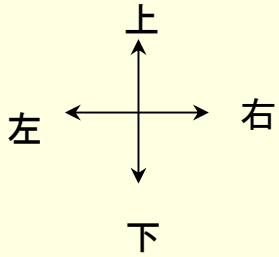


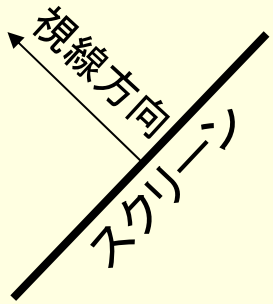
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	0	1	1	0
左	0	0	0	0	0	0	0	0	0
右	1	0	0	1	0	0	1	1	0

: 第２段レンダリング済

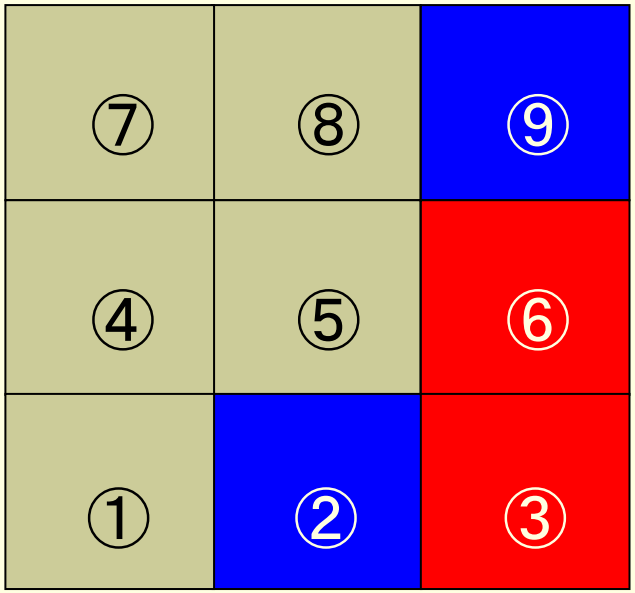
: 第１段レンダリング中

: 未処理





オブスタクルフラグの使用例

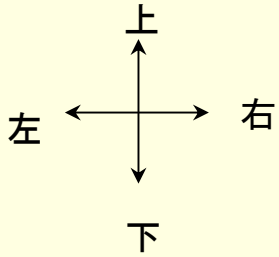


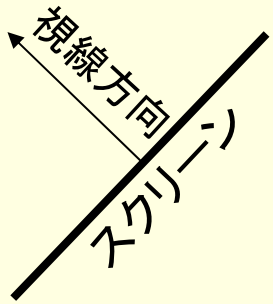
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	0	1	1	0
左	0	0	0	0	0	0	0	0	0
右	1	0	0	1	0	0	1	1	0

: 第２段レンダリング済

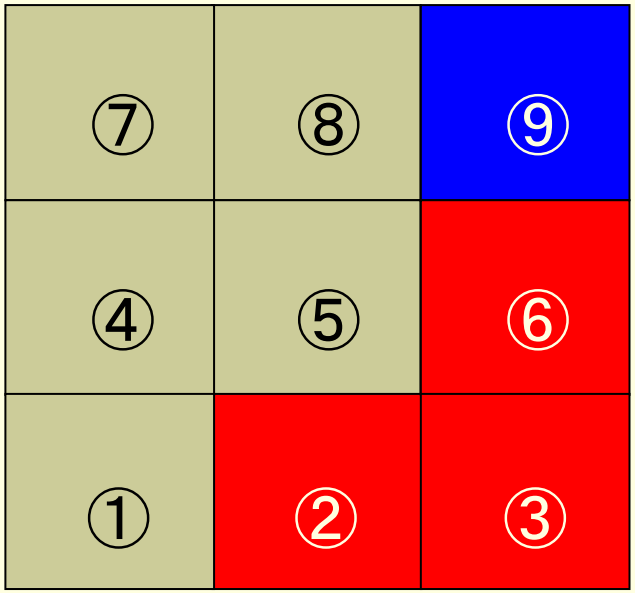
: 第１段レンダリング中

: 未処理

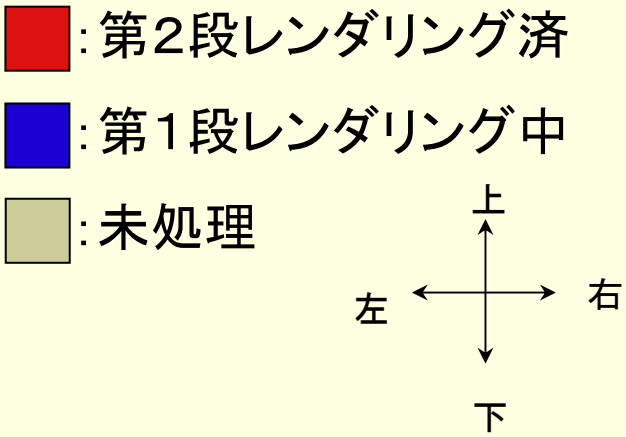




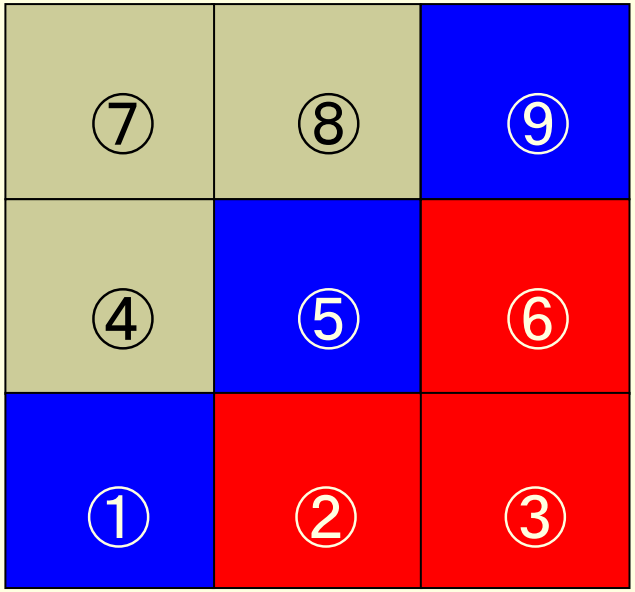
オブスタクルフラグの使用例



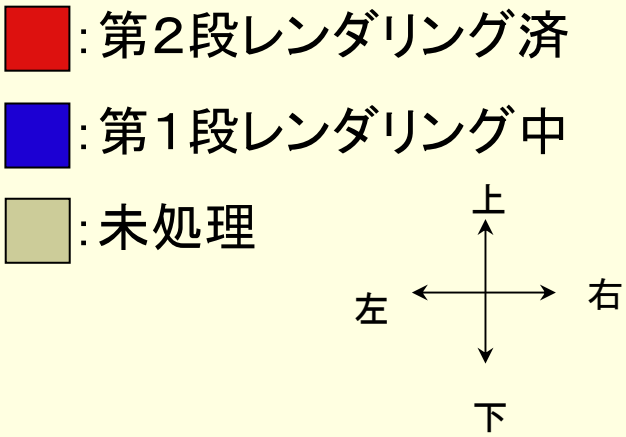
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	0	0	1	1	0
左	0	0	0	0	0	0	0	0	0
右	0	0	0	1	0	0	1	1	0



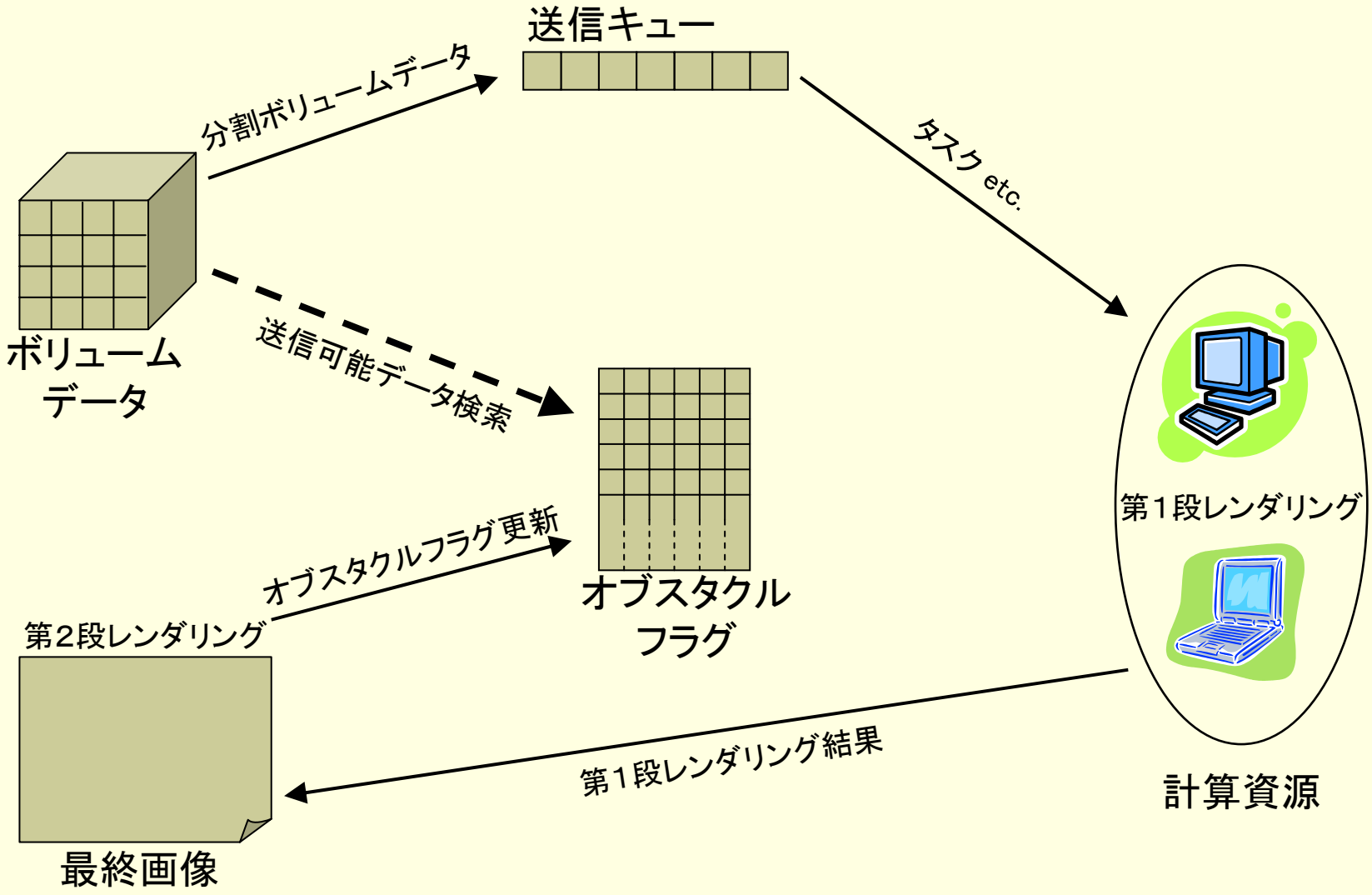
オブスタクルフラグの使用例



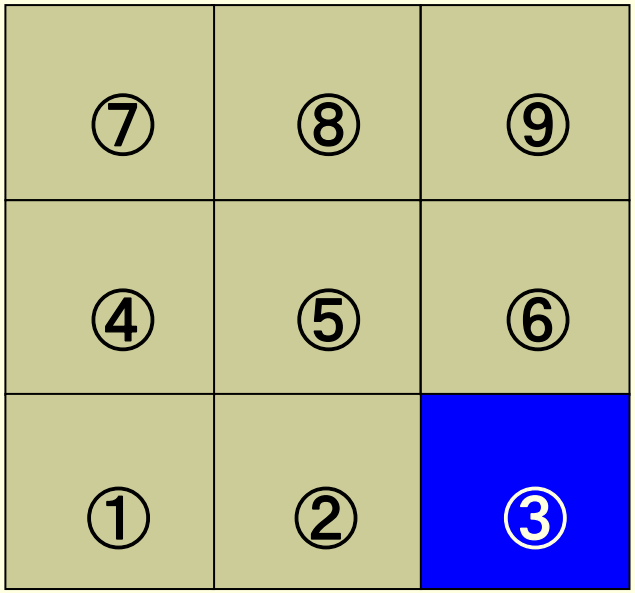
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	0	0	1	1	0
左	0	0	0	0	0	0	0	0	0
右	0	0	0	1	0	0	1	1	0



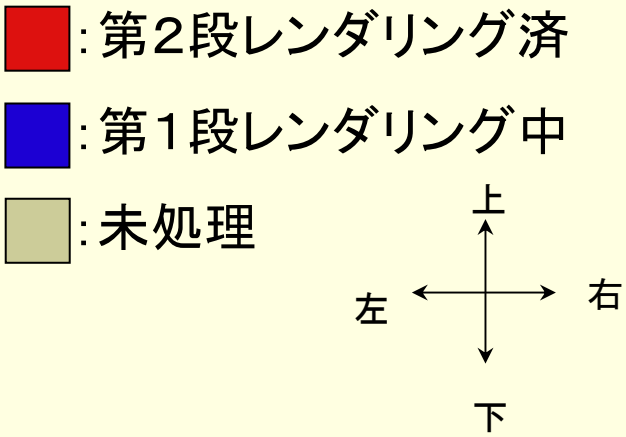
提案手法による処理1

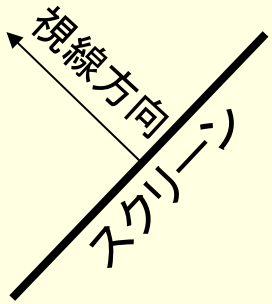

送信キュー
分割ボリュームデータ
タスク etc.
ボリューム
データ
送信可能データ検索
オブスタクル
フラグ
オブスタクルフラグ更新
第2段レンダリング
第1段レンダリング
第1段レンダリング結果
計算資源
最終画像
参照

計算資源の有効利用



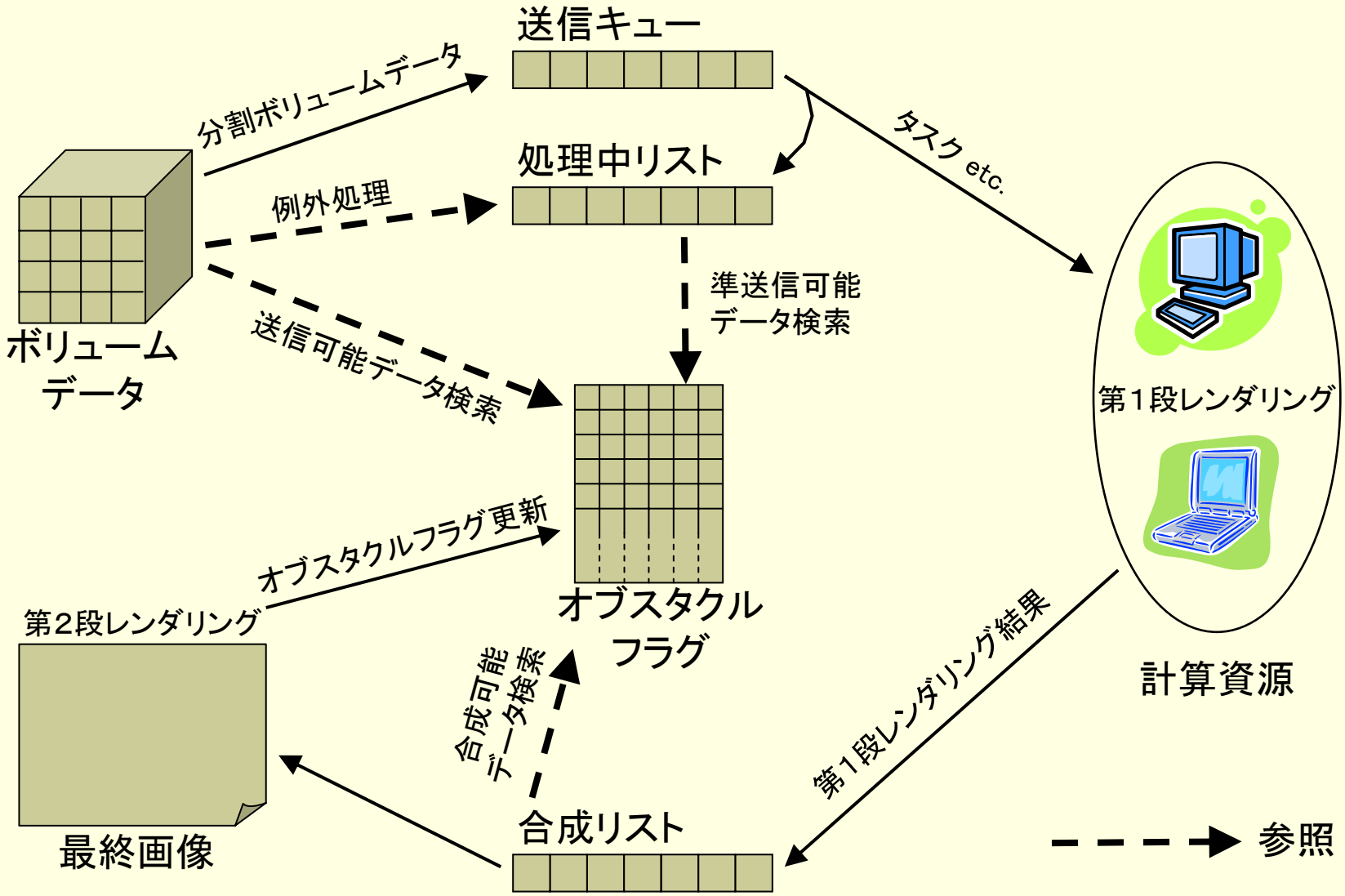
	①	②	③	④	⑤	⑥	⑦	⑧	⑨
上	0	0	0	0	0	0	0	0	0
下	0	0	0	1	1	1	1	1	1
左	0	0	0	0	0	0	0	0	0
右	1	1	0	1	1	0	1	1	0





計算資源が複数空いていても
タスクが1つしか投入できない

提案手法による処理2


送信キュー
分割ボリュームデータ
処理中リスト
タスク etc.
例外処理
準送信可能
データ検索
ボリューム
データ
送信可能データ検索
第1段レンダリング
オブスタクル
フラグ
オブスタクルフラグ更新
第2段レンダリング
計算資源
合成可能
データ検索
第1段レンダリング結果
合成リスト
最終画像
参照

シミュレーションによる比較実験

- 提案手法の有用性の検証

提案手法（提案手法による処理1）

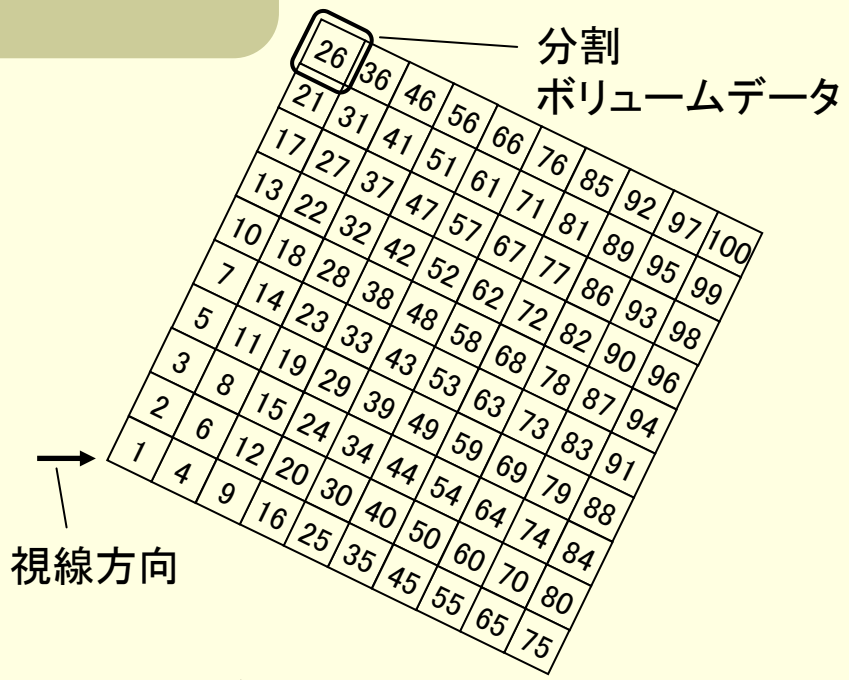
・動的に更新されるオブスタクルフラグを参照し
投入するタスクを決定

V.S.

**シーケンシャル
タスクスケジューリング**

（基本的な分散処理）

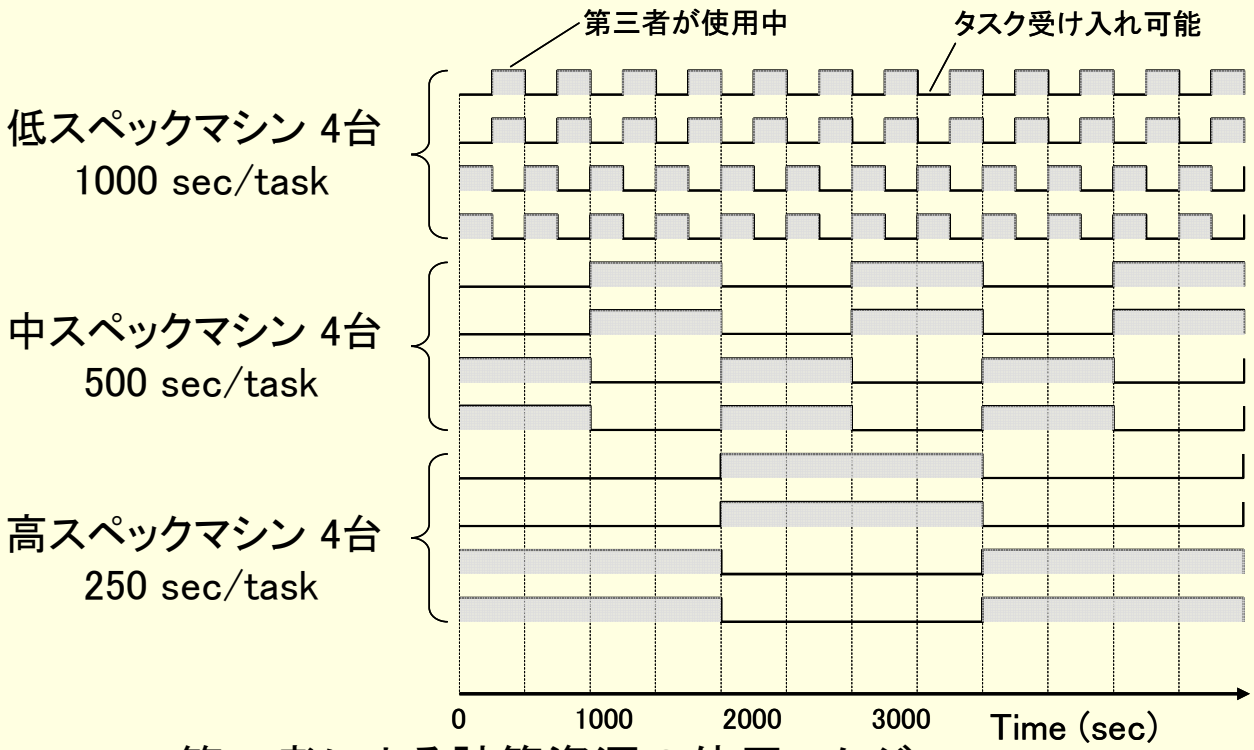
・初期に決定した可視性に基づく
順序に従い投入するタスクを決定



2次元データで実験
数字は可視性に基づく順序

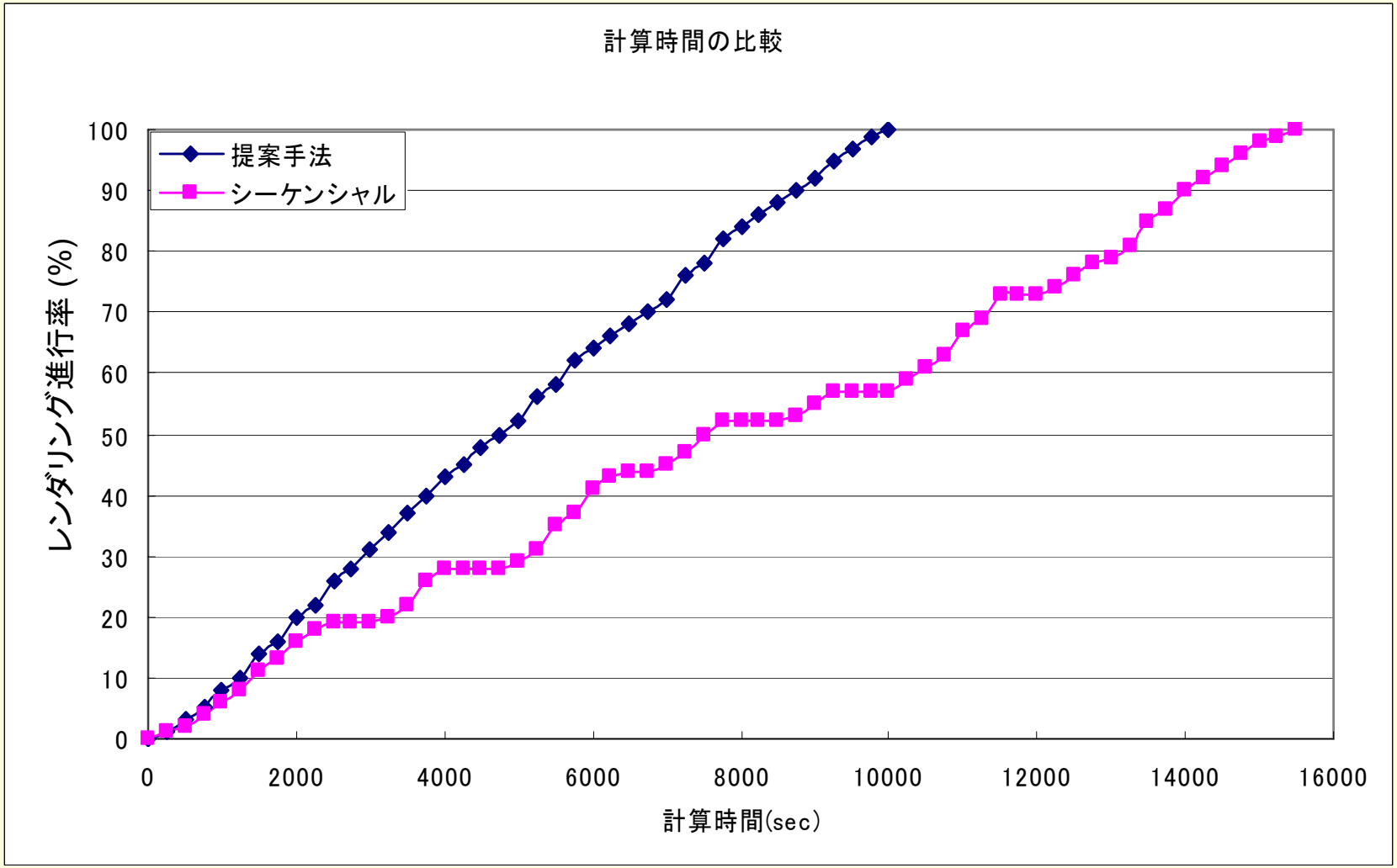
シミュレーション環境

- グリッドコンピューティングの環境を仮定
  - 計算資源は不均一
  - 定期的に第三者により使用される
  - 第三者使用中は計算資源の提供は停止される

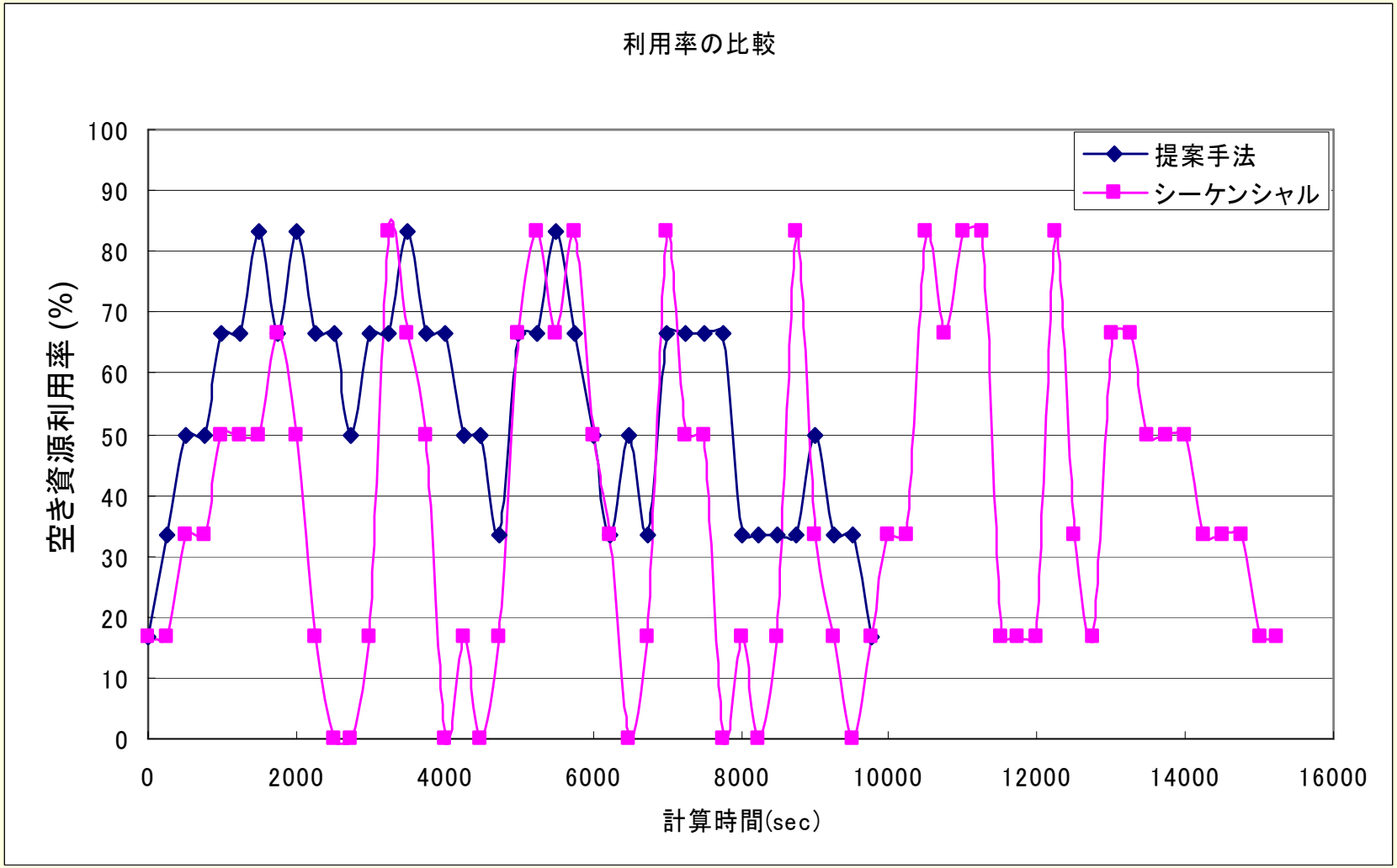


第三者による計算資源の使用スケジュール

シミュレーション結果：時間比較



シミュレーション結果：利用率比較



結果

- シミュレーションの比較

	提案手法	シーケンシャル
計算時間	10,000 sec	15,500 sec
計算資源 平均利用率	54.2 %	38.4 %

- 実際に構築したグリッドシステムでのレンダリング時間

グリッドコンピューティング (6CPU)※1	1CPU※2
113 sec	439 sec

※1　Power PC 2GHz～1.6GHz
※2　Power PC 1.8GHz

スクリーンサイズ：1800^2　ボリュームデータ：1024^3

まとめ

グリッドコンピューティングを用いた
ボリュームレンダリングの高速化のために

- オブスタクルフラグを提案
- オブスタクルフラグを用いた動的なタスクを投入
- シミュレーションにより提案手法の有用性を検証

今後の課題

- 実際のシステムを用いた実験
  - 分割数・環境・ボリュームサイズ etc.
- 計算資源の能力を考慮したタスクの割り振り