創造情報学輪講

千葉研究室
M1
夏澄彦

自己紹介

- 千葉研 M1（Core Software Group)
- 22歳
- 中国生まれ（4歳の時に日本へ）
- 上海→町田→府中→国分寺
- 横浜国立大学電子情報工学科
- 卒論：モバイルセンサネットワーク

仮想ばね力学に基づく モバイルセンサネットワークの最適配置

想定環境

人が立ち入ることができない環境に対して、動力を持ったセンサノードを自律的に展開させることにより、未知の障害物の存在する領域にセンサネットワークを構築する

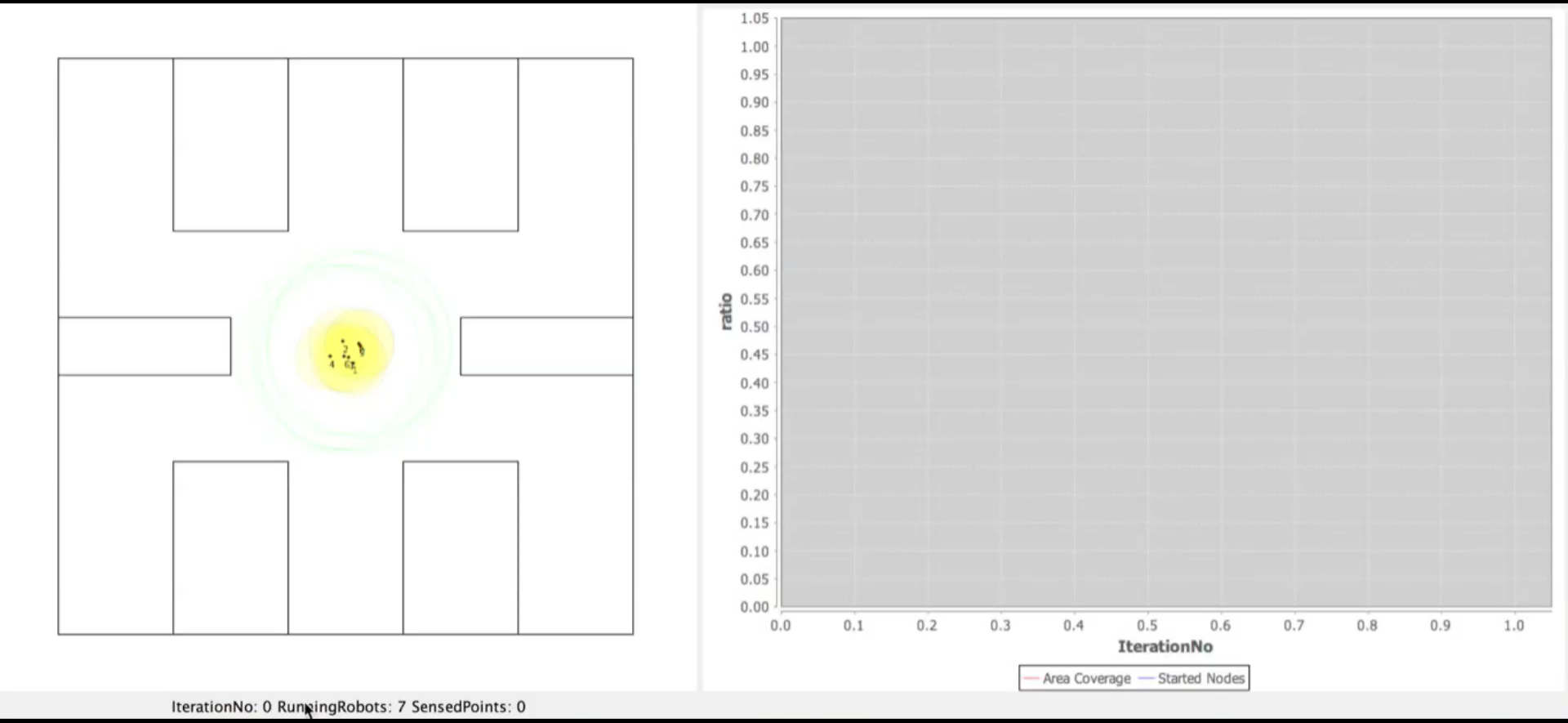
既存研究

センサ間に仮想的なばねを想定することで、センサ間の距離を適切に保ち、ネットワークの連結性を保ったまま、カバレッジ率を最大化できる

提案

- ばねの自然長をセンサの残電力に応じて減少させることにより、通信路を増やすのと同時に、ノードの寿命が切れた時の連結性を高める
- ノードを一気に展開するのではなく、シンクノード周辺の仮想的なばねの力関係に応じて、適応的にノードを展開させることによりネットワークの寿命を伸ばす

仮想ばね力学に基づく
モバイルセンサネットワークの最適配置



千葉研の課題

言語系

- Java拡張
- コンパイラ・インタプリタ
- DSL

HPC系

- MPI
- CUDA
- X10

千葉研の課題

言語系

- Java拡張
- コンパイラ・インタプリタ
- DSL

HPC系

- MPI
- CUDA
- X10

千葉研の課題

言語系

- Java拡張

MPIなんてつまんないよね
by 某千葉研の某助教

HPC系

- ~~MPI~~ → JavaScriptによる並列処理
- CUDA
- X10

コンパイラ・インタプリタ

目的

プログラミング言語の基盤技術への理解を深める

やること

Lex&Yacc&LLVMを利用したStone言語のインタプリタを実装する
（参考図書：2週間でできる! スクリプト言語の作り方）

Stone言語

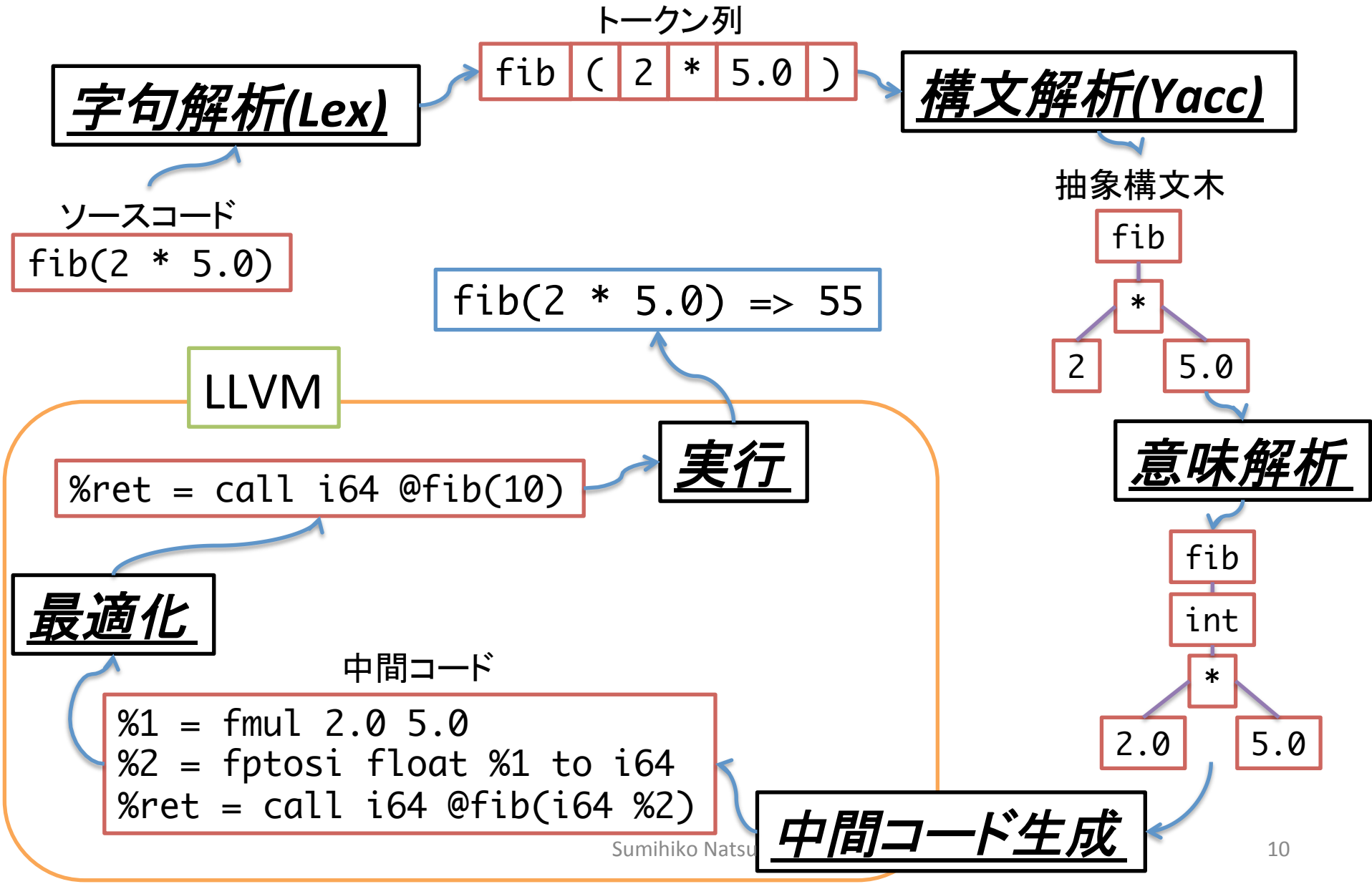
簡単なプログラミング言語

- ✓ 四則演算
- ✓ if-else式
- ✓ 変数（型付き：整数、小数）
- ✓ 関数

例）フィボナッチ数列の計算

```
def fib(n:int):int {
  if (n > 1) {
    fib(n - 1) + fib(n - 2)
  } else {
    n
  }
}
fib(10)
```

Overview


トークン列
fib (2 * 5.0)
字句解析(Lex)
構文解析(Yacc)
ソースコード
fib(2 * 5.0)
抽象構文木
fib
*
2
5.0
fib(2 * 5.0) => 55
LLVM
実行
意味解析
%ret = call i64 @fib(10)
fib
int
*
2.0
5.0
最適化
中間コード
%1 = fmul 2.0 5.0
%2 = fptosi float %1 to i64
%ret = call i64 @fib(i64 %2)
中間コード生成

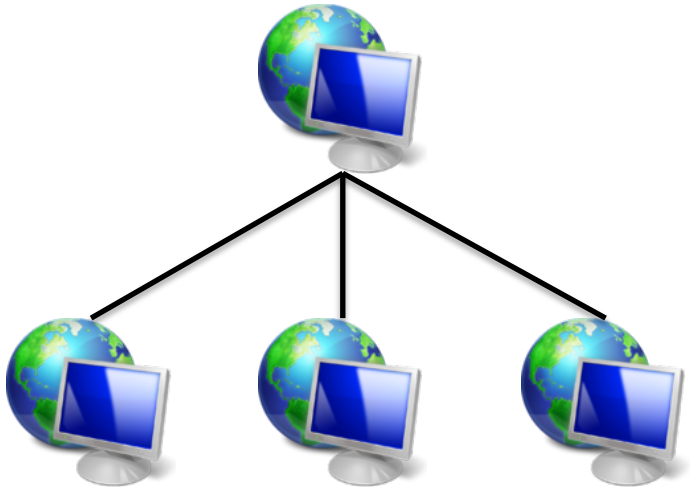
JavaScriptによる並列処理

目的

分散コンピューティングへの理解を深める

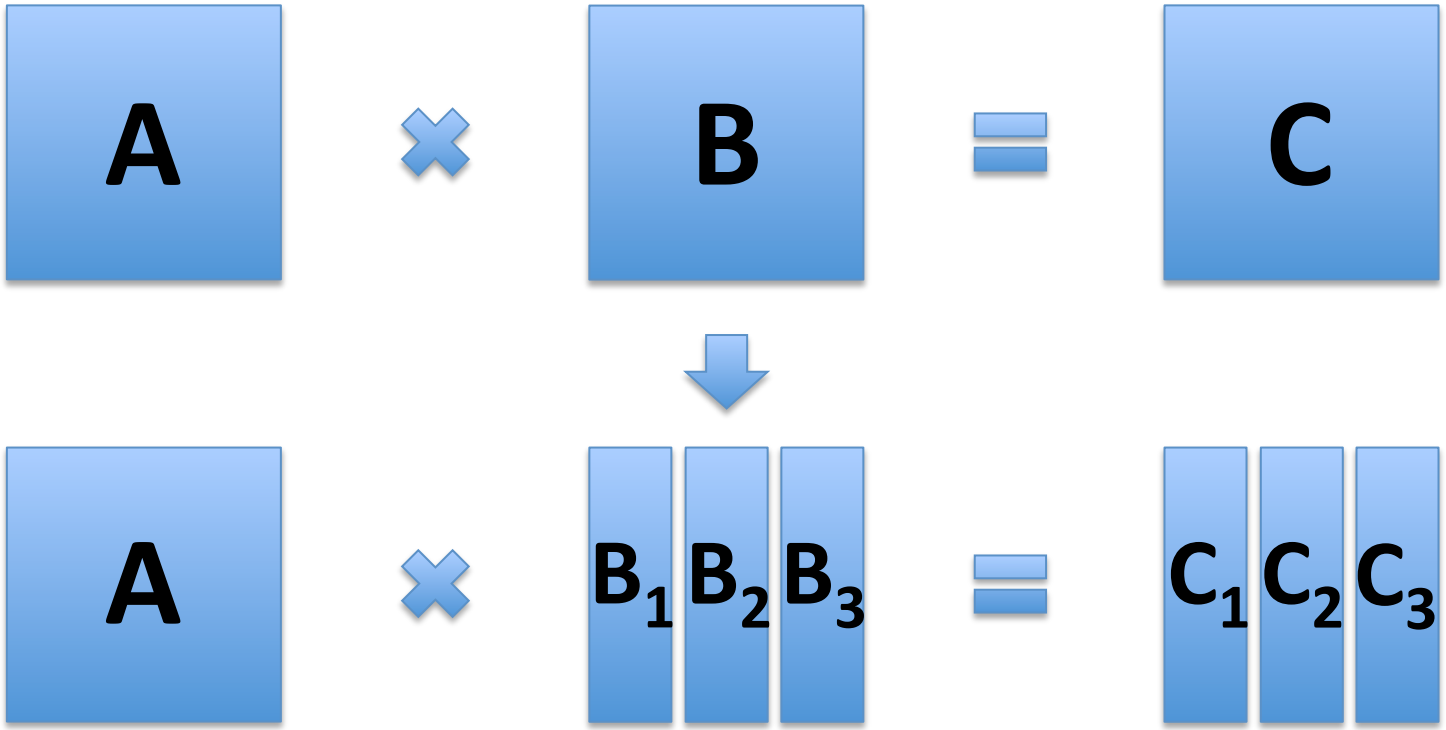
やること

JavaScriptの様々な技術を利用し、2次元空間上の粒子を
衝突シミュレーションを複数のブラウザにより並列化する



予備実験

- 行列積の分散・並列処理
  - 一つのブラウザによる並列処理
  - 複数のブラウザによる分散処理



一つのブラウザによる行列積の並列処理

- 実験方法
  - 1024×1024の行列積
  - シングルスレッド
  - CPU
  - GPU
- 使用技術
  - WebWorker (CPU)
  - RiverTrail (GPU)
- 実験環境

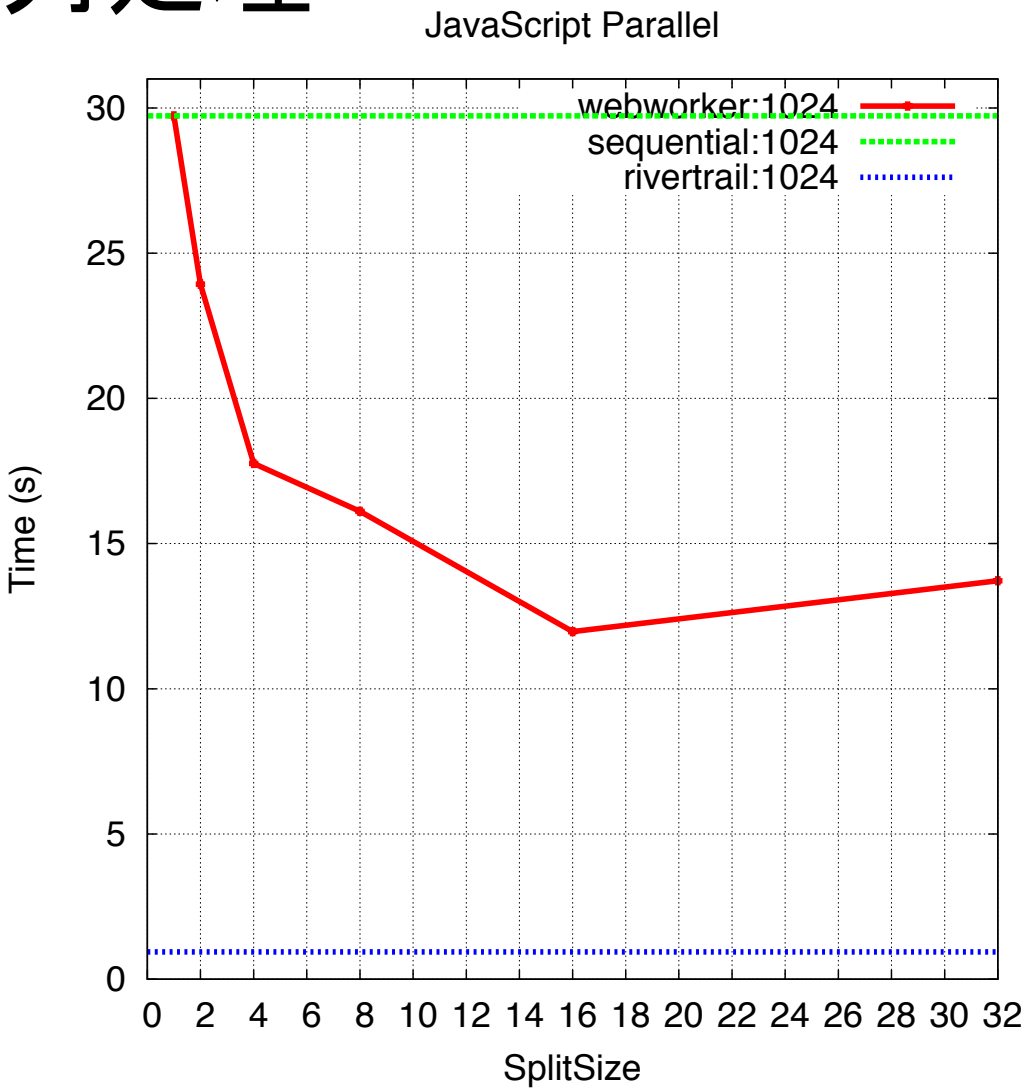
  MacBook Pro Retina

  Processor: 2.7GHz Intel Core i7

  Memory:　16GB

  Graphics: NVIDIA GeForce GT 650M

  Firefox 21.0



一つのブラウザによる行列積の並列処理

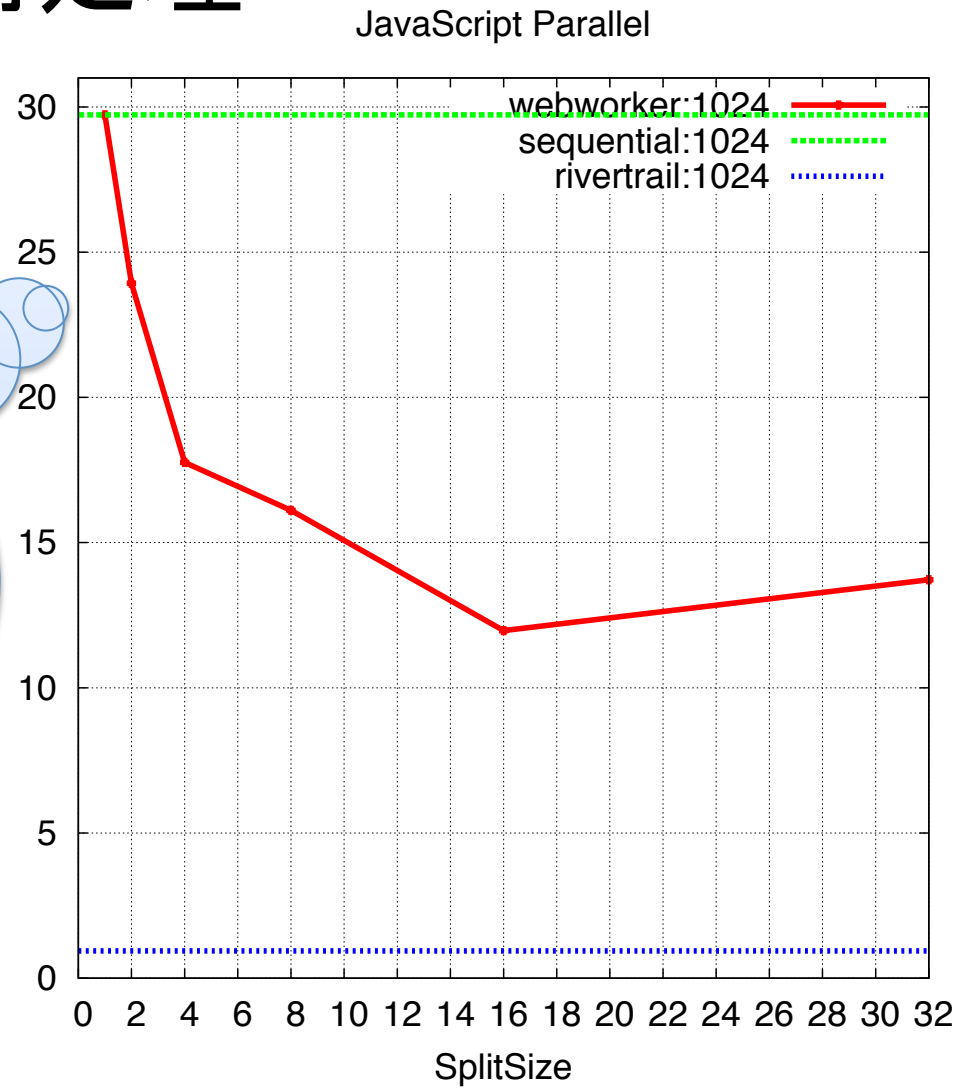
- 実験方法
  - 1024×1024の行列積
  - シン

シングルスレッドと比べて

マルチスレッドでは
最大で半分以下

GPUを使用した場合には
20分の1以下

Graphics: NVI GeForce GT 650M
Firefox 21.0



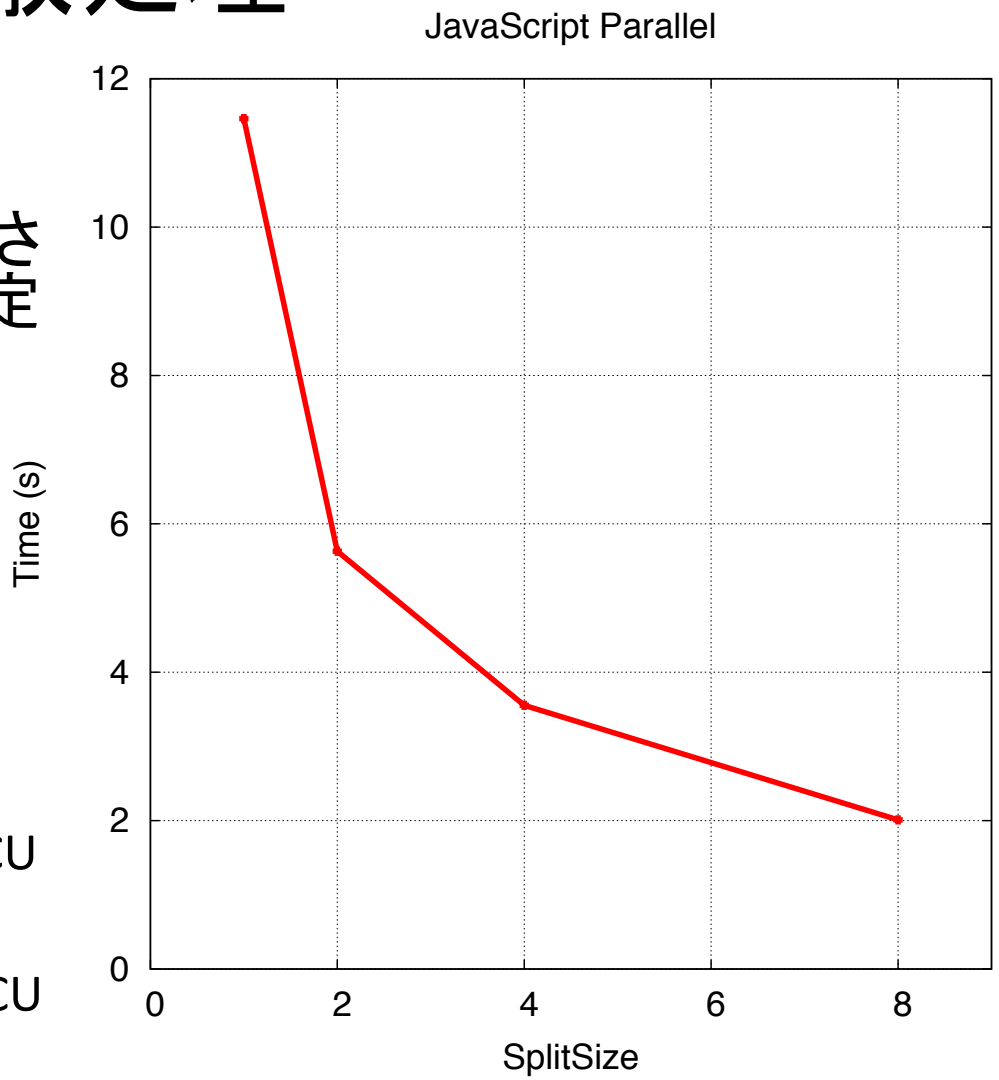
複数のブラウザによる行列積の分散処理

- 実験方法
  - 256×256の行列積
  - 分散数を1～8台と変化させた時の実行時間を測定
- 使用技術
  - WebSocket（通信）
  - Node.js（サーバ）
- 実験環境

  edubase Cloud CentOS5.6
  - マスターサーバ（1台）

    M1 large Memory7.5GB 4ECU
  - ブラウザ（1～8台）

    M1 small Memory1.7GB 1ECU



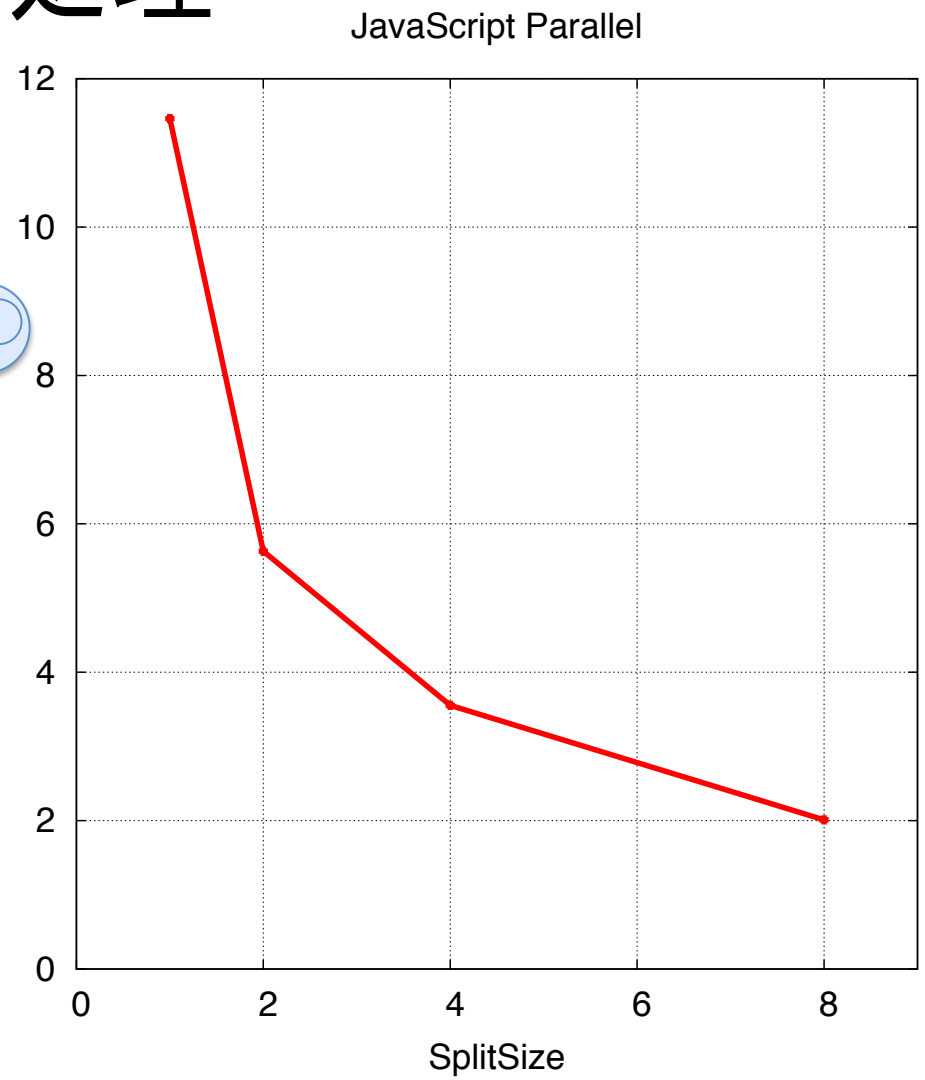
複数のブラウザによる行列積の分散処理

- 実験方法
  - 256×256の行列積
  - 分散化さ
  - .6
  - 台）
  - ory7.5GB 4ECU
  - ブラウ 1～8台）
    M1 small Memory1.7GB 1ECU

1台の場合と比べて

8台で分散処理すると

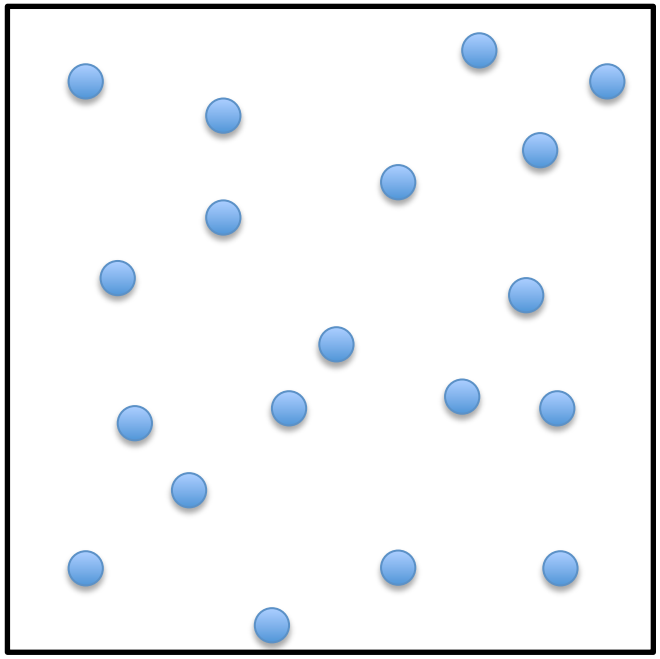
5分の1以下



粒子の衝突シミュレーション

目標

- 粒子間及び粒子と境界間の衝突をシミュレーションする
- 次の衝突時間を逐次計算し、粒子の移動・衝突を繰り返す
- ブラウザを利用し、リアルタイムにレンダリングする

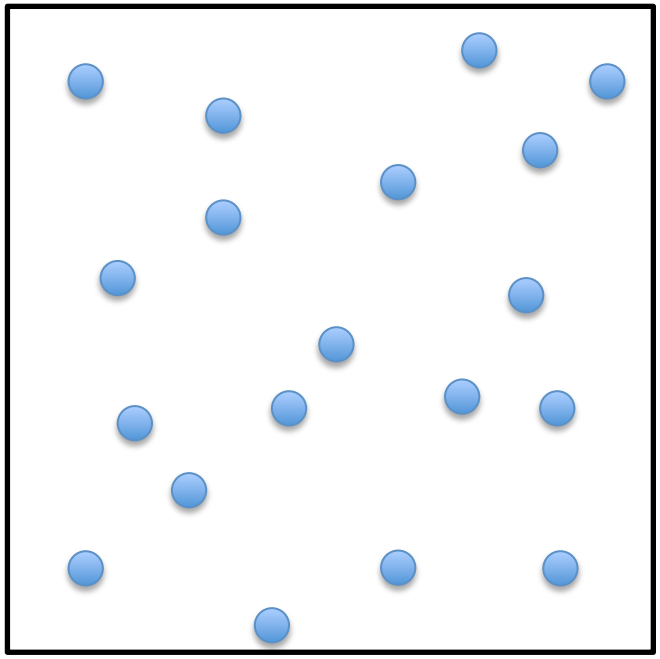


剛体の衝突シミュレーション

目標

- 粒子間
- 次の衝
- ブラウザを利用し、

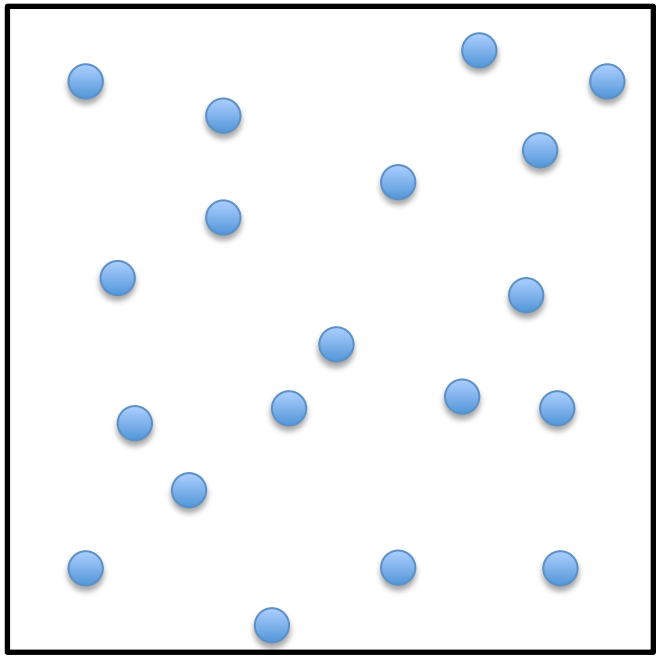
粒子は大きさを持たない
by 某M.I.さん



粒子の衝突シミュレーション

目標

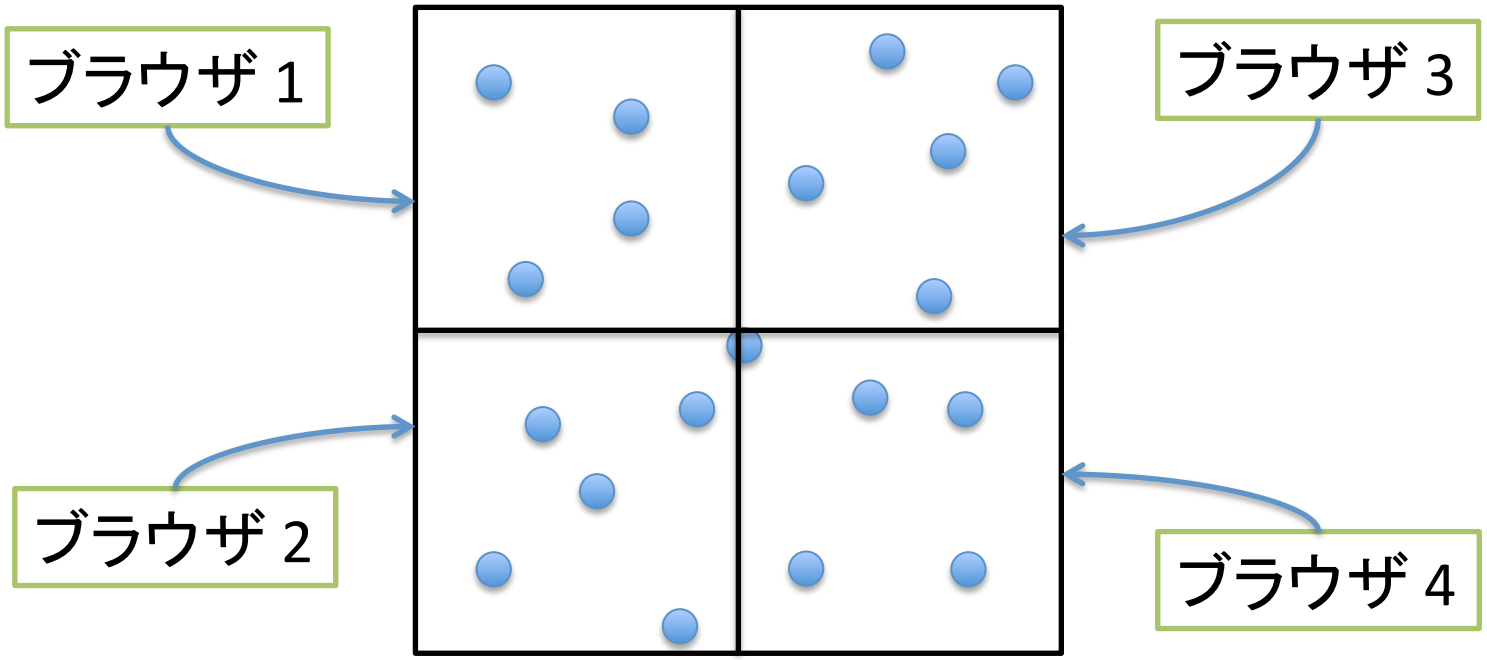
- 粒子間及び粒子と境界間の衝突をシミュレーションする
- 次の衝突時間を逐次計算し、粒子の移動・衝突を繰り返す
- ブラウザを利用し、リアルタイムにレンダリングする



並列分散処理のための手法

手法

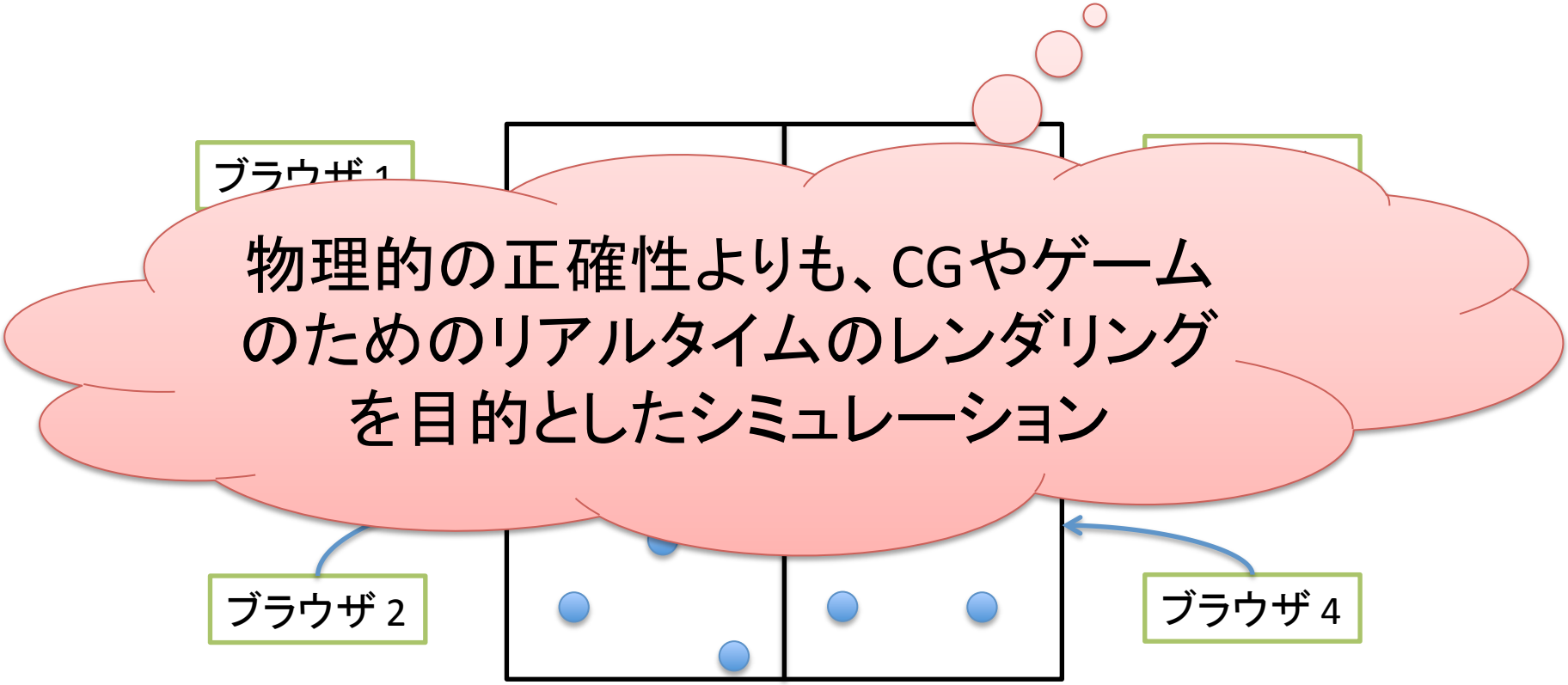
- 領域を分割し、各ブラウザに処理を分散させる
- 粒子の移動が速くなり過ぎないように、上限速度を設ける



並列分散処理のための手法

手法

- 領域を分割し、各ブラウザに処理を分散させる
- 粒子の移動が速くなり過ぎないように、上限速度を設ける



通信量の削減

問題点

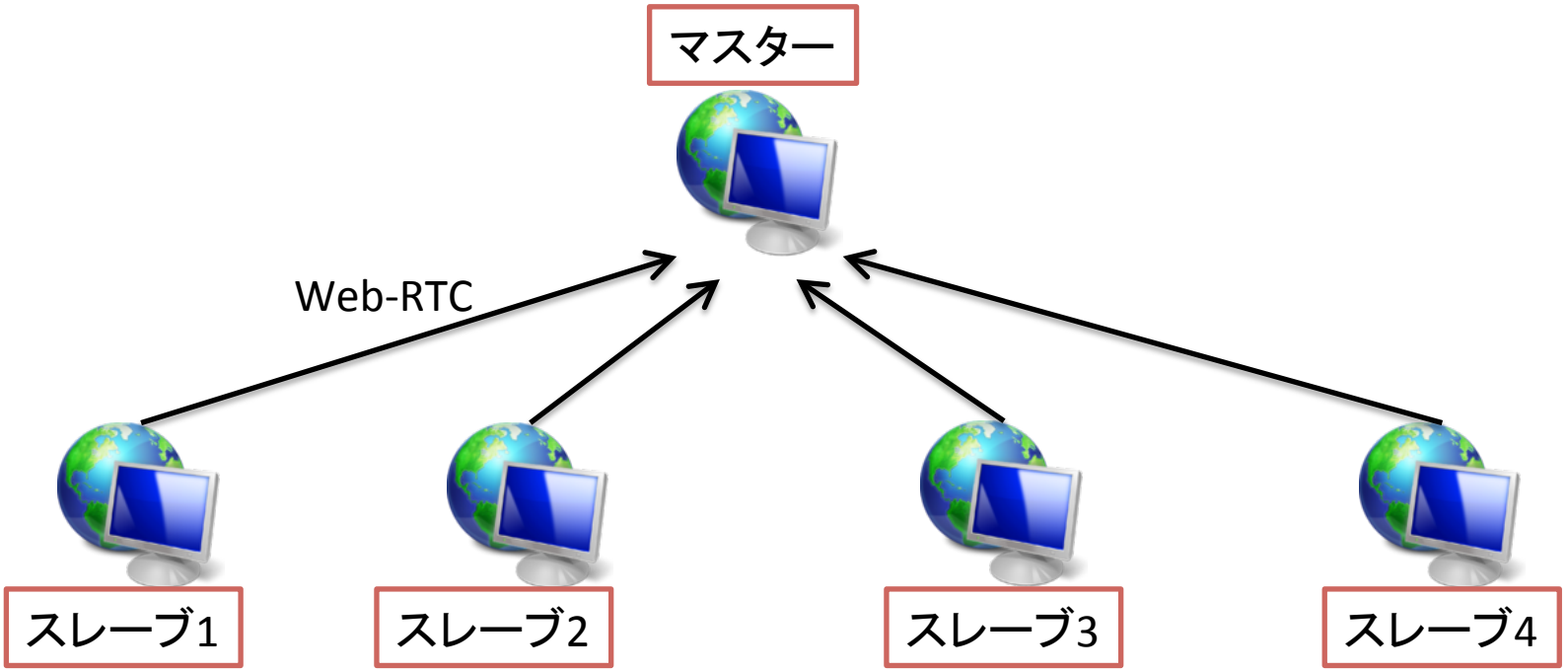
- 粒子が各計算領域を移動するので、隣接領域と通信する
- リアルタイムにレンダリングするため、結果を全て一つのブラウザへ送信する

→ 計算よりも通信がボトルネック

解決法

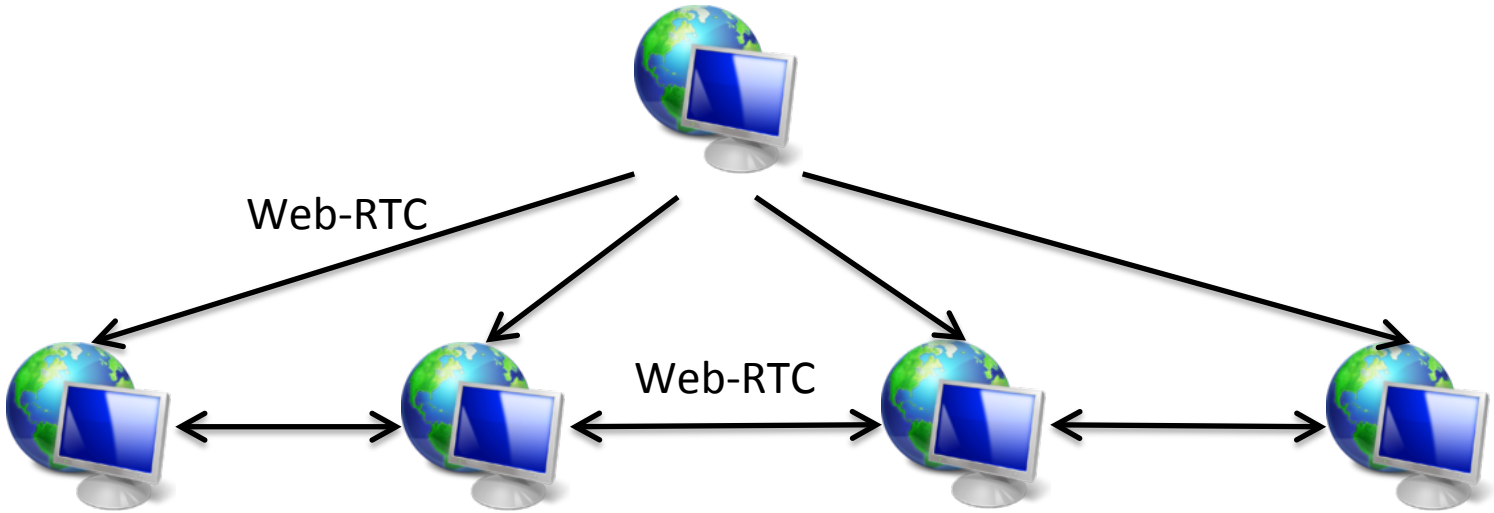
- WebRTCによりブラウザ間でP2Pで通信
  → マスターサーバに通信が集中しない
- 担当領域に加えて袖領域の粒子状態を保持
  → 一定時間通信しなくても計算可能
- 衝突時間とその2物体のIDのみマスターに送信し、マスターでは受信した衝突イベントを再生することで粒子状態を再現
  → 全ての粒子状態を送信せずにすみ、通信量が減る

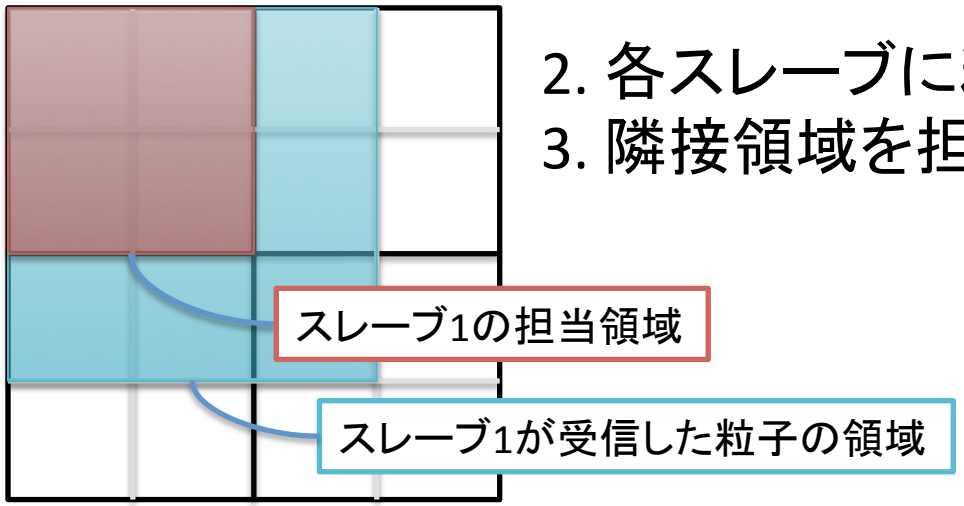
シュミレーションフロー



1. マスターに各スレーブが接続

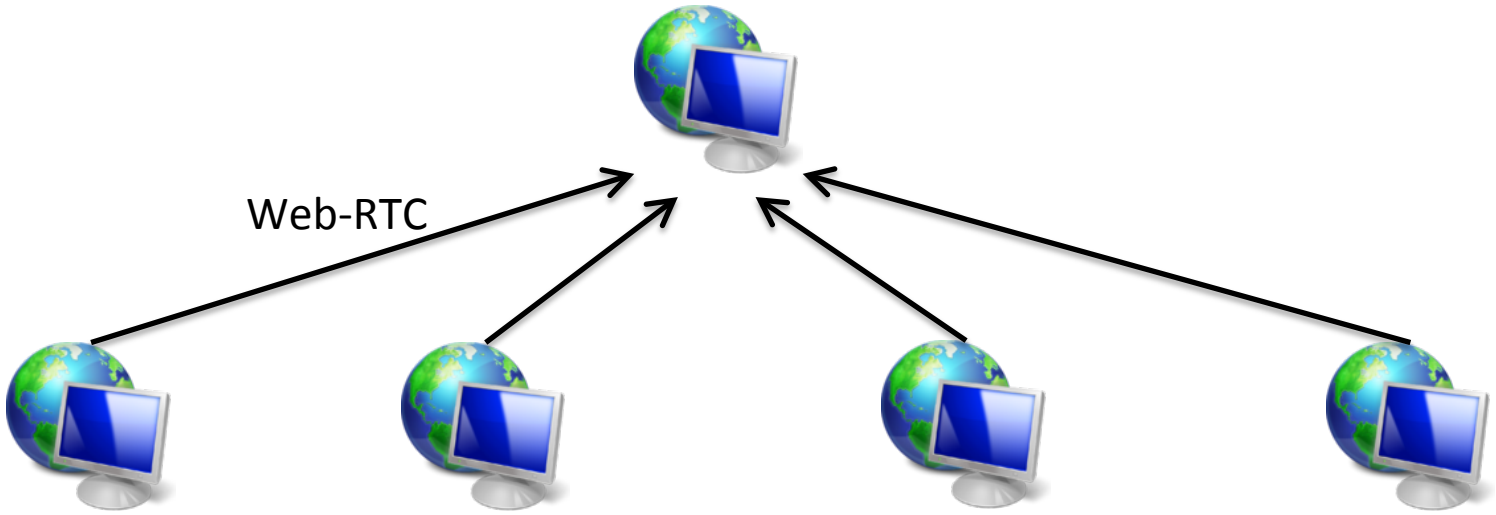
シュミレーションフロー

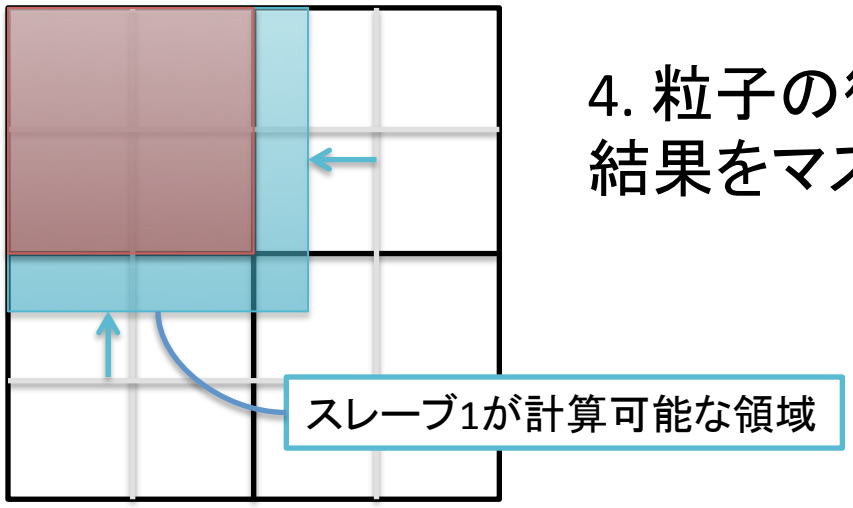




2. 各スレーブに粒子の初期状態を送信
3. 隣接領域を担当している他のスレーブと接続

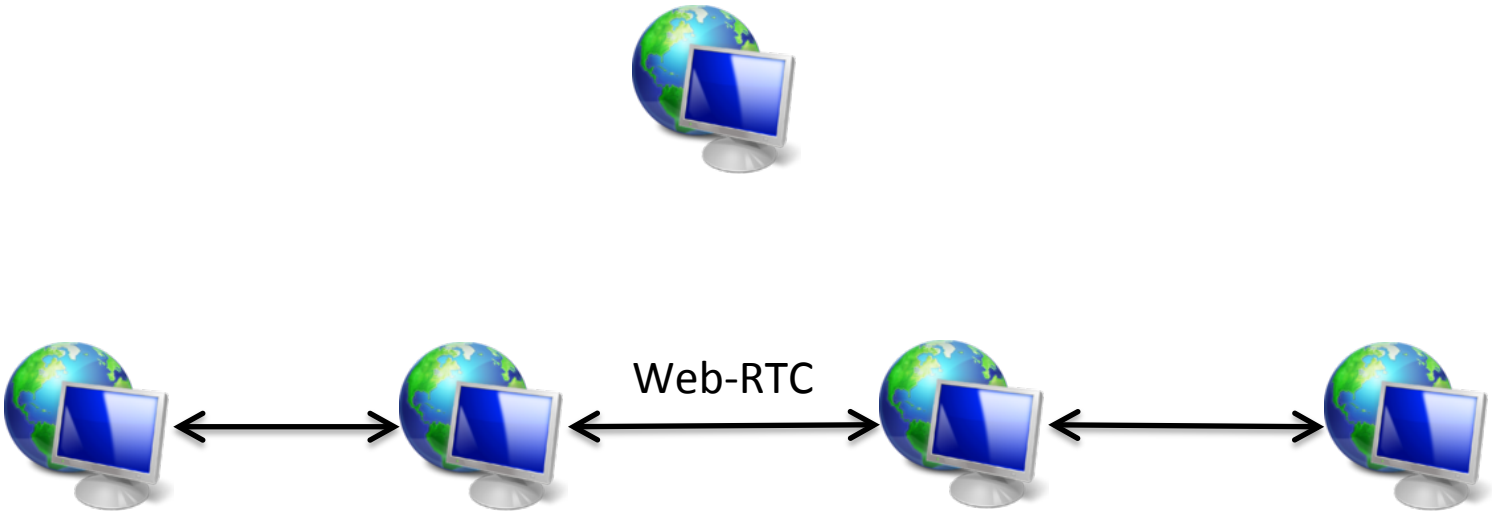
シュミレーションフロー

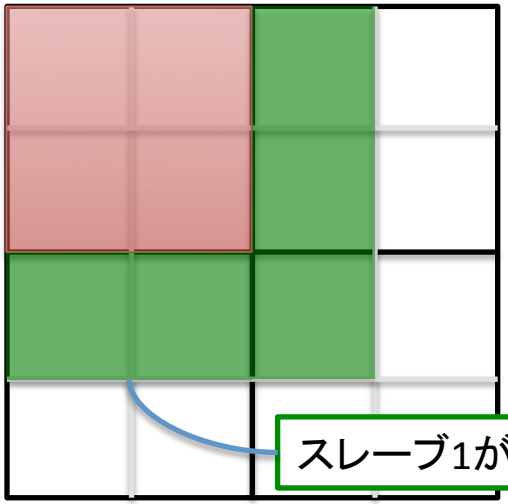




4. 粒子の衝突を計算し、シミュレーション結果をマスターへ送信

シュミレーションフロー

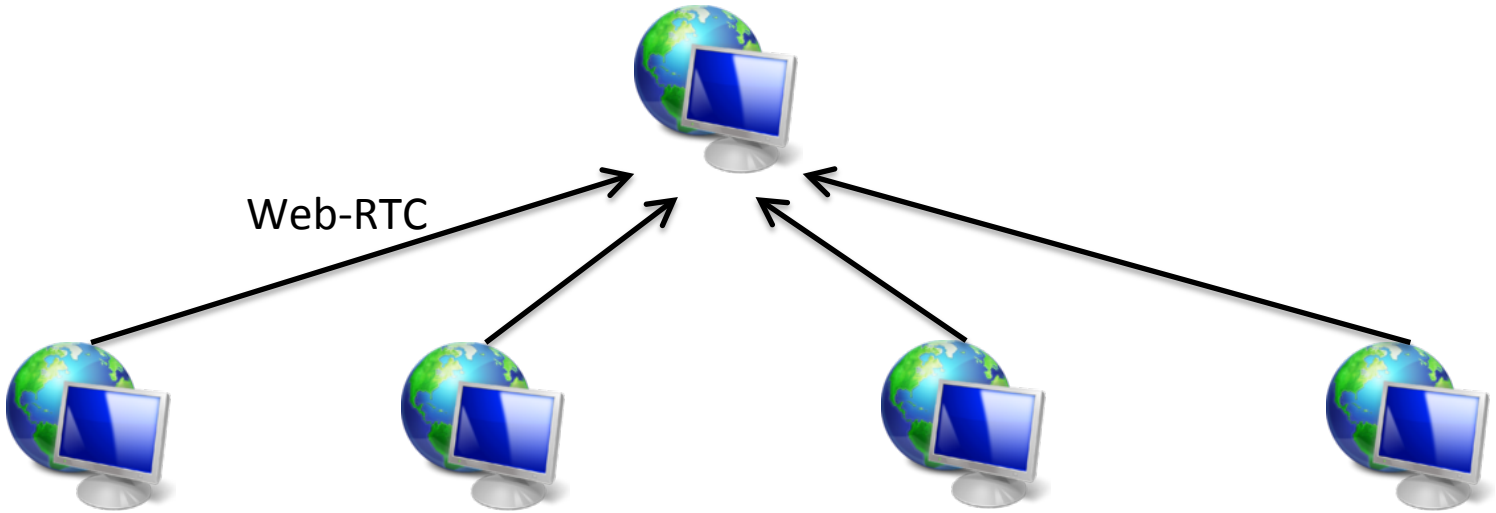


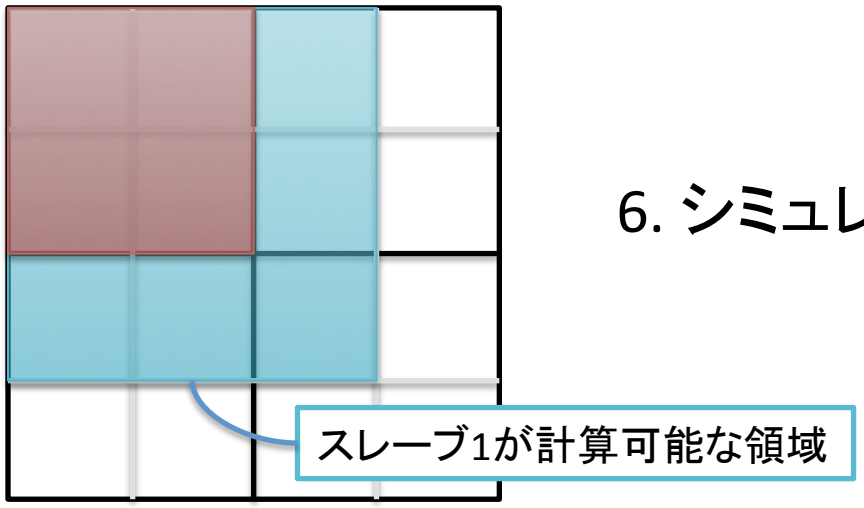


5. 隣接領域を担当している他スレーブから粒子状態を受信

スレーブ1が他スレーブから粒子状態を受信する領域

シュミレーションフロー





6. シミュレーションを再開する

シミュレーション

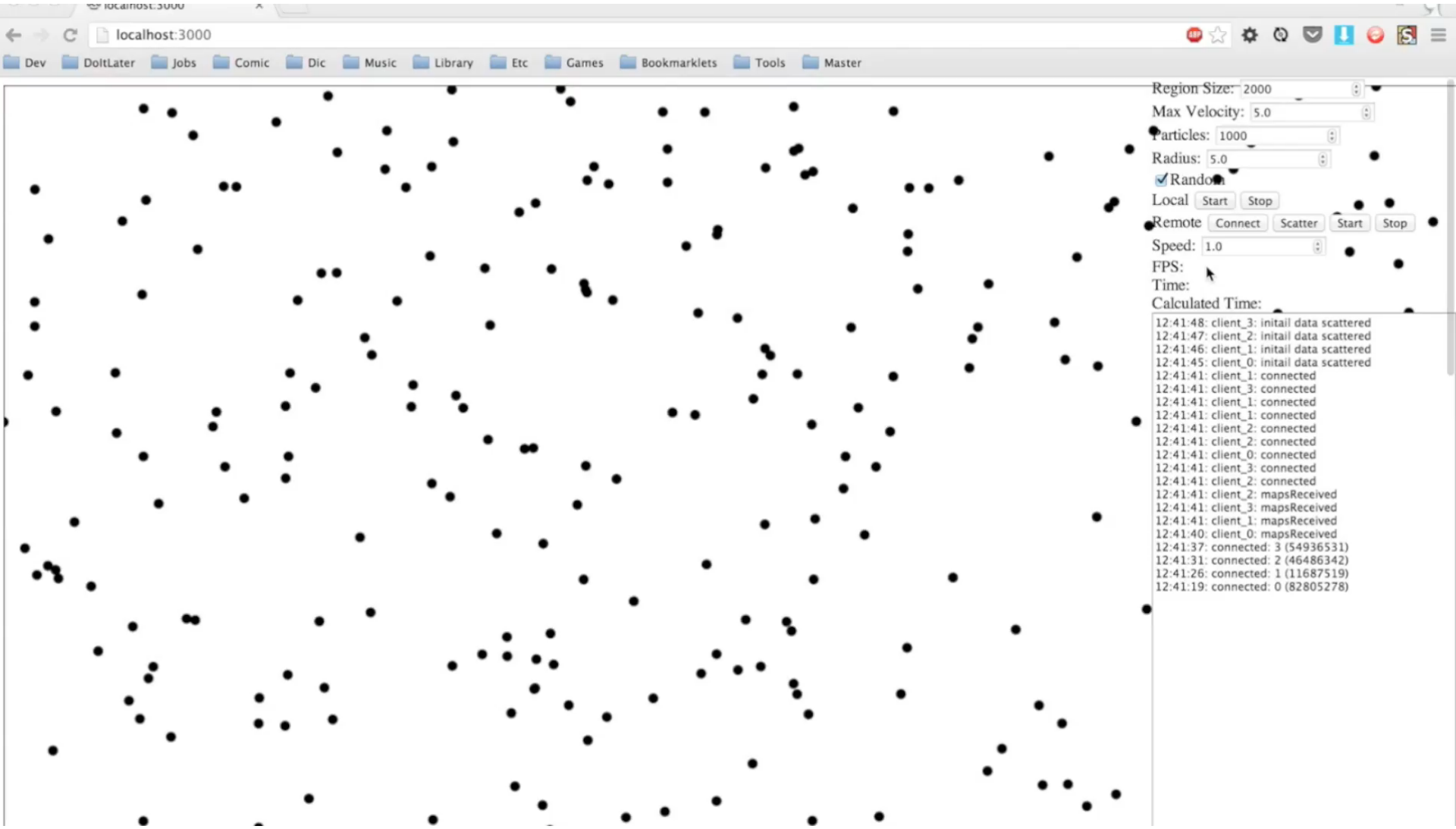
- Amazon EC2のM1 smallインスタンス5台

  Memory: 1.7GB, 1ECU

- マスター（１台）
  - Node.jsサーバ
    - 各ブラウザ間でP2P通信するためのIPアドレスの管理
  - マスターブラウザ
- スレーブ（４台）
  - スレーブブラウザ

- 粒子数: 1000
- 粒子の上限速度: 5.0
- 粒子の半径: 5.0
- 領域の大きさ: 2000

シミュレーション


localhost:3000
Dev
DoItLater
Jobs
Comic
Dic
Music
Library
Etc
Games
Bookmarklets
Tools
Master
Region Size: 2000
Max Velocity: 5.0
Particles: 1000
Radius: 5.0
Random
Local Start Stop
Remote Connect Scatter Start Stop
Speed: 1.0
FPS:
Time:
Calculated Time:
12:41:48: client_3: initail data scattered
12:41:47: client_2: initail data scattered
12:41:46: client_1: initail data scattered
12:41:45: client_0: initail data scattered
12:41:41: client_1: connected
12:41:41: client_3: connected
12:41:41: client_1: connected
12:41:41: client_1: connected
12:41:41: client_2: connected
12:41:41: client_2: connected
12:41:41: client_0: connected
12:41:41: client_3: connected
12:41:41: client_2: connected
12:41:41: client_2: mapsReceived
12:41:41: client_3: mapsReceived
12:41:41: client_1: mapsReceived
12:41:40: client_0: mapsReceived
12:41:37: connected: 3 (54936531)
12:41:31: connected: 2 (46486342)
12:41:26: connected: 1 (11687519)
12:41:19: connected: 0 (82805278)

シミュレーション結果

120秒分の衝突シミュレーション

１台でシミュレーション

228s

４台でシミュレーション

44s

注）複数台でシミュレーションする場合には粒子の初期状態を各ブラウザに展開してから時間の計測を開始する

修論に向けて（検討中）

- JavaScriptによる並列・分散処理を簡易化するためのフレームワーク
- 動的型付け言語と静的型付け言語のハイブリッド言語（Rubyで変数の型指定とか、、、）

シミュレーション結果

120秒分の衝突シミュレーション

１台でシミュレーション

228s

４台でシミュレーション

44s

注）複数台でシミュレーションする場合には粒子の初期状態を各ブラウザに展開してから時間の計測を開始する

JavaScriptの様々な技術

- 通信
  - Ajax

    最も標準的なHTTP通信を利用したブラウザ・サーバ間の通信技術
  - WebSocket

    通信を全ての一つのコネクション上で行うことにより、HTTPの欠点を改善した技術
  - Web-RTC

    ブラウザ間のP2P通信を行う技術
- 並列処理
  - WebWorker

    JavaScriptでマルチスレッド処理を行う技術
  - RiverTrail、WebCL

    JavaScriptからGPUなどへアクセスできる技術