EXTREME GLOBAL COUPLER 利用マニュアル

Hideyuki Jitsumoto

jitumoto@gsic.titech.ac.jp

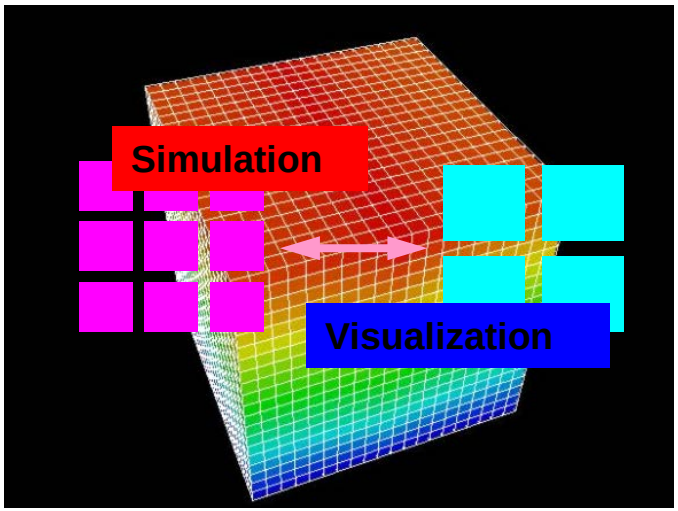
Co-Researcher

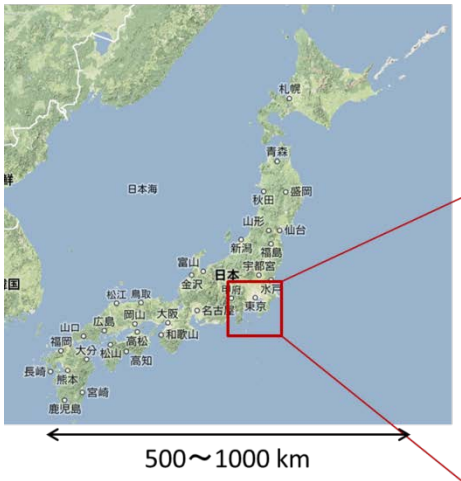
Taizo Kobayashi, Masaharu Matsumoto
Shin'ichiro Takizawa, Shin'ichiro Miura
Kengo Nakajima, Toshihiro Hanawa

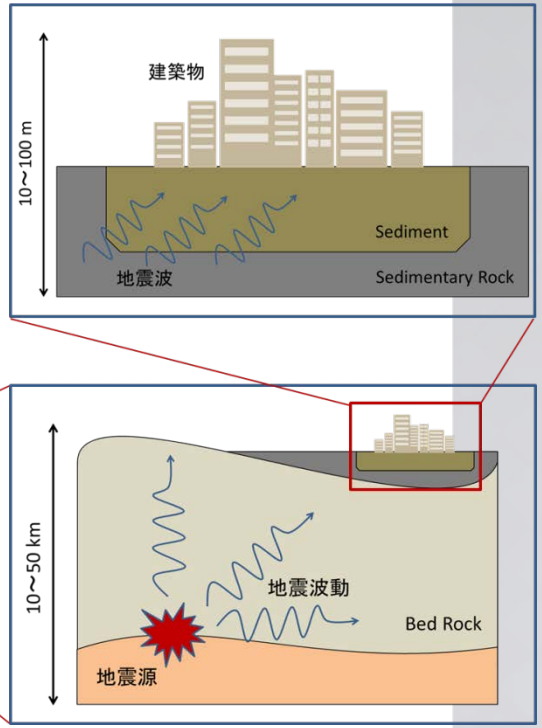
EXTREME GLOBAL COUPER
背景

連成計算アプリケーション

- 複数のアプリで構成されるアプリ
  - Pre-Post連携（計算＋可視化処理など）
  - マルチスケール
  - マルチフィジクス
- さらなる高精度・広範囲
  - より多くの資源要求

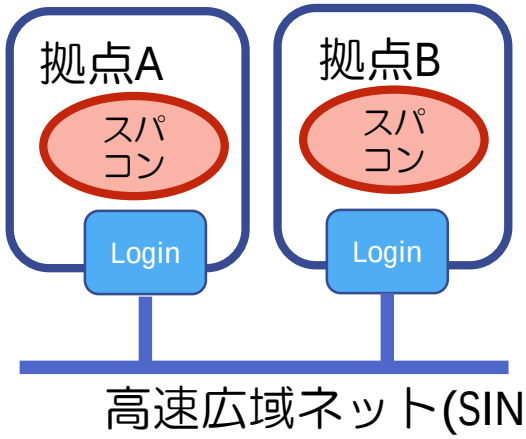


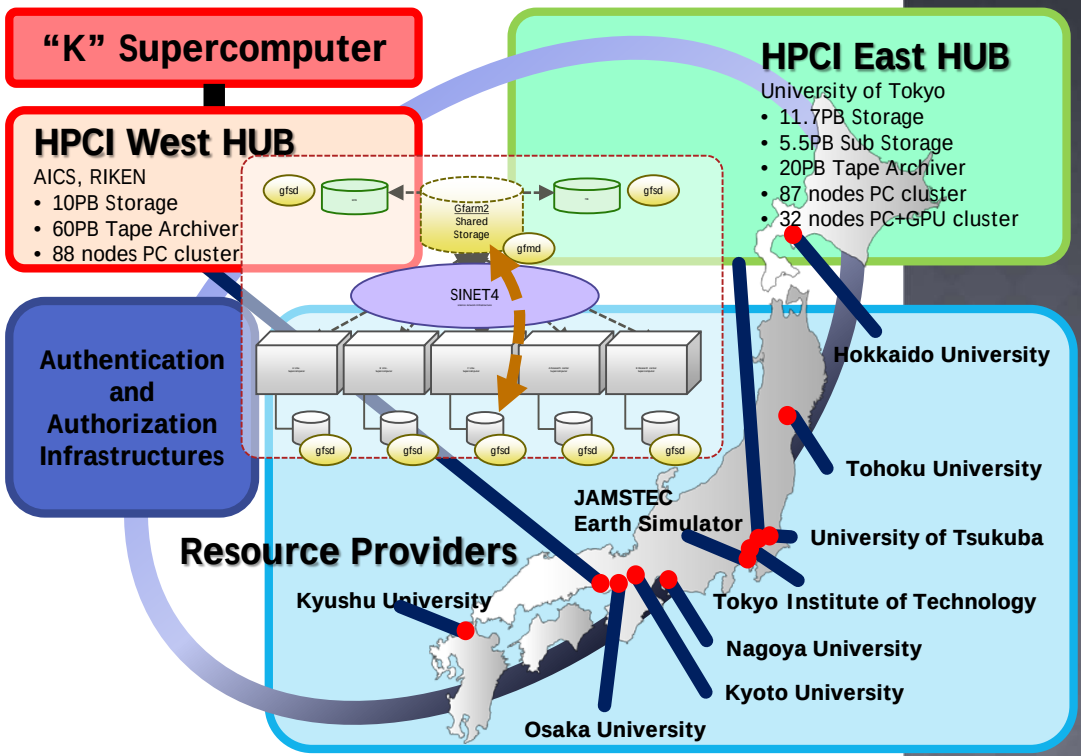




HPCI:革新的ハイパフォーマンス・コンピューティング・インフラ

- 京を中心に複数の拠点の持つスパコンを連携させる環境
  - SSO
  - 共有ストレージ
  - SINET4
- 拠点毎のポリシ管理
  - 空きポートやサービス





計算資源の大規模確保 ～多拠点連携アプリケーション

- JHPCN/HPCI での複数の計算拠点間協調
- 学術および商用クラウドの成熟
- マルチスケール構造や可視化を伴うアプリケーションの一般化

- 広域に分散した計算機資源を有効に活用する
  - 連成するアプリ間での負荷の違いや疎結合性を前提とした、複数拠点での計算資源の確保
  - ライセンス制限や拠点間に分散配置されたデータによる実行制限への対応

既存拠点間連携MWの問題点

- NAREGI
  - グリッドミドルウェアパッケージ
    - SSO と VO →一部 HPCI にも利用
    - 共有ファイルシステムと高効率データ転送
    - 資源管理 (Super Scheduler, Information Service)

  →巨大で複雑／拠点に合わせた追加開発

  →構築拠点間のポリシ合意が欠かせない
- RENKEI-POP/VPE
  - グリットコンピューティング用のサブセット
    - 高効率データ転送
    - VM+VPNによるワンサイト化

  →拠点間合意によるサービス構築が必要

拠点間連携MW 必要要件

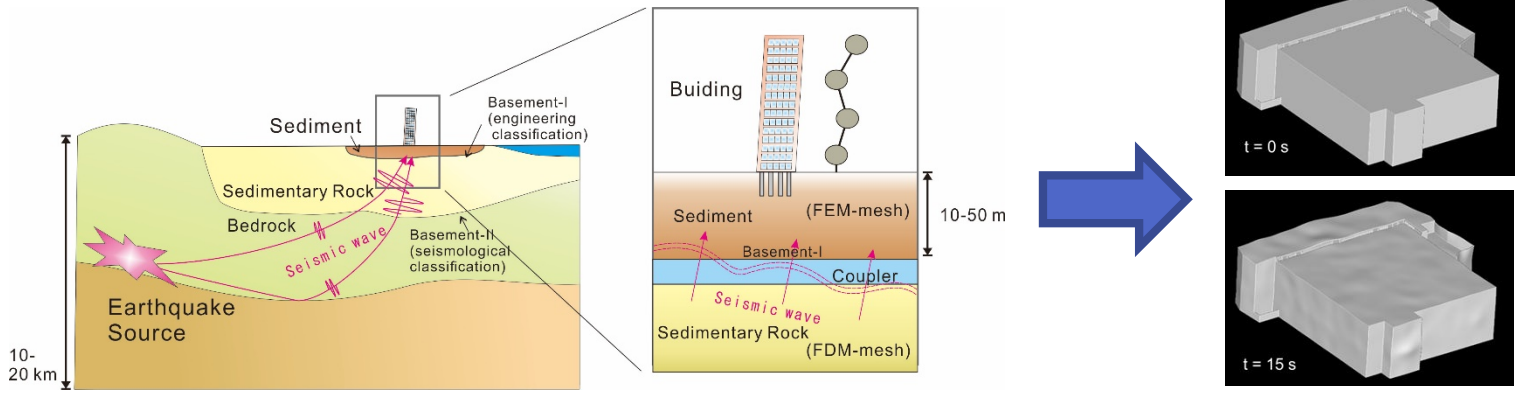
- 多くの拠点間で利用可能な通信路の作成（通信）
  - 計算ノードはプライベート空間に存在することが多い
  - 全計算ノードをグローバルに置くことはセキュリティ／台数的にも不可能
- 連成アプリケーションが同時実行できない状況での通信（ジョブ投入）
  - それぞれの拠点のスケジューラは連携していない
  - 連携するには追加開発が必要になってしまう
- 上記をユーザ権限で達成可能な仕組み
  - サービスとして各拠点が実行するには運用ポリシに関する政治的な問題を解決する必要がある

研究の目的

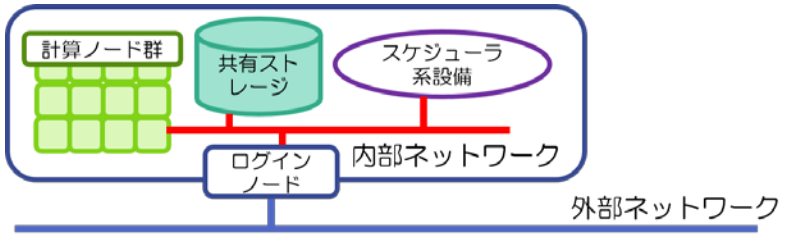
- 実アプリを多拠点で利用した際の影響の検証と効率化
  - 多拠点利用による性能向上が期待できるシナリオの確認
  - 多拠点で効率よくジョブを実行する際の最適化パラメータの確認
  - 連成計算を構成する簡易な手法
- 多拠点連成計算アプリを構成・実行するための導入障壁の低いフレームワークの構築
  - 各拠点の運用ポリシを阻害せず、政治的なコストを最小にできる環境構築手法

連成アプリケーションと想定する実行環境

- マルチスケール構造
- 前・後処理を含むアプリケーション
  - 一方向通信



- 環境
  - ログインノードと計算ノード間にデータ居通信を行える設備を持つ

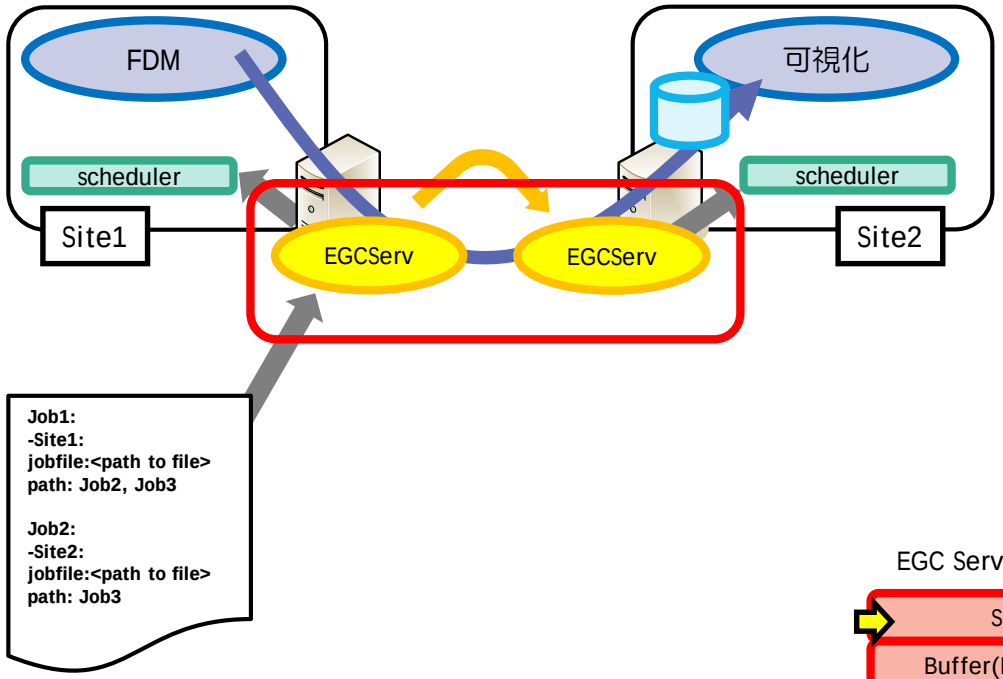


EXTREME GLOBAL COUPER
設計・実装

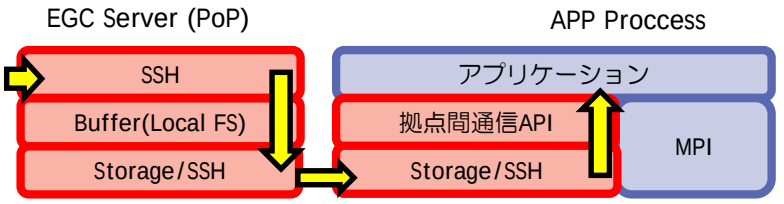
フレームワーク設計

- 要件
  - 多くの拠点間で利用可能な通信路の作成（通信）
  - 連成アプリケーションが同時実行できない状況での通信（ジョブ投入）
  - 上記をユーザ権限で達成可能な仕組み
- 方針
  - SSH や共有ストレージなど、スパコンセンターに必須な well-known サービスのみで構成
  - ログインノード上に通信をリダイレクションするプロセスを配置しメッセージを管理
    - 受信側での通信データ保存により、ジョブの同時投入を不要に
    - サイト内通信では可能であれば共有ファイルシステムを経由し、それが持つ並列性を利用する
  - 環境間差異はユーザが手動で対応する
    - ジョブ投入ファイルフォーマット

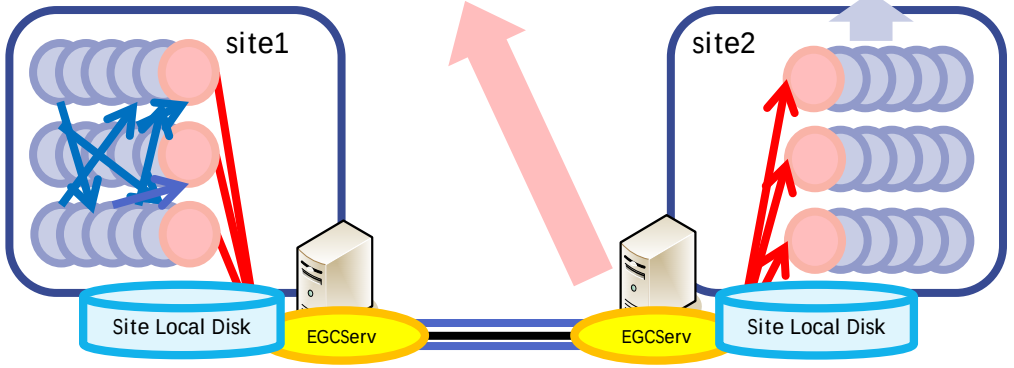
コンポーネント概観



- EGC Server
  - ログインノード上のメッセージ管理サーバ
  - 各拠点へのジョブ投入



- 拠点間通信API
  - Send/Recv
  - ソース毎の送信順の保持



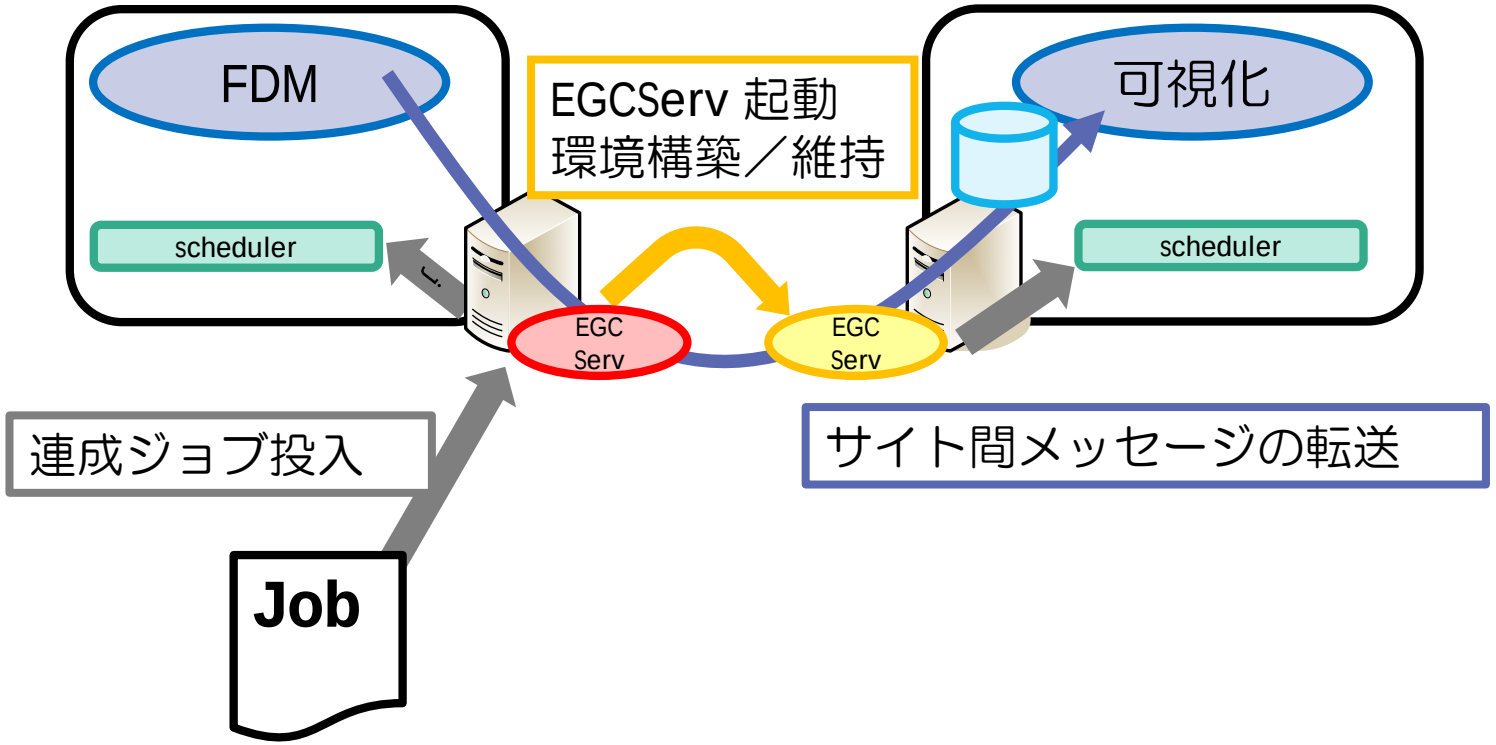
コンポーネント(1/2) - EGC サーバ

- ログインノード上の管理サーバ
  - アプリケーションを拠点毎のスケジューラ適切に提出
  - 拠点間メッセージをプライベートネットワークにリダイレクト
    - 連携アプリが同時に起動してない場合のメッセージバッファリング
  - 拠点間で互いに監視し、必要に応じて起動し合う

コンポーネント(2/2) - 拠点間通信ライブラリ

- 拠点間の P2P 通信を提供する
  - 自拠点内の EGC サーバへのブロッキング送受信
  - メッセージ識別タグ(送信元ID, 受信先ID)
    - 連成されるそれぞれのアプリ内でプログラマが任意に作成する整数値ID
    - アプリで利用される MPI Rank をそのまま使うことが可能
    - 同じタグが付いたメッセージについて順序を保つ

EGC 機能概観


FDM
EGCServ 起動
環境構築／維持
可視化
scheduler
scheduler
EGC
Serv
EGC
Serv
連成ジョブ投入
サイト間メッセージの転送
Job

環境構築

- 1サイトからの実行による環境構築
- SSH トンネリングを利用したサーバ間転送

EGC_HOSTNUM=%d	以降に記述される拠点数
EGC_PACKET_SIZE=%d	通信のパケットコンテナサイズ
EGC_HOSTNAME%d=%s	拠点のログインノード名
EGC_SSHPORT%d=%d	拠点のログインノードに**SSH**接続するポート番号
EGC_HOSTPORT%d=%s	拠点のログインノードで待ち受ける**(LISTEN)**ポート番号
EGC_PROXYPORT%d=%s	接続者が**SSH**プロキシとして用意する**HOSTPORT**のプロキシポート
EGC_JOBSUBMIT%d=%s	拠点毎のジョブ投入命令
EGC_CONFIG%d=%s	拠点毎のこの設定ファイルの位置

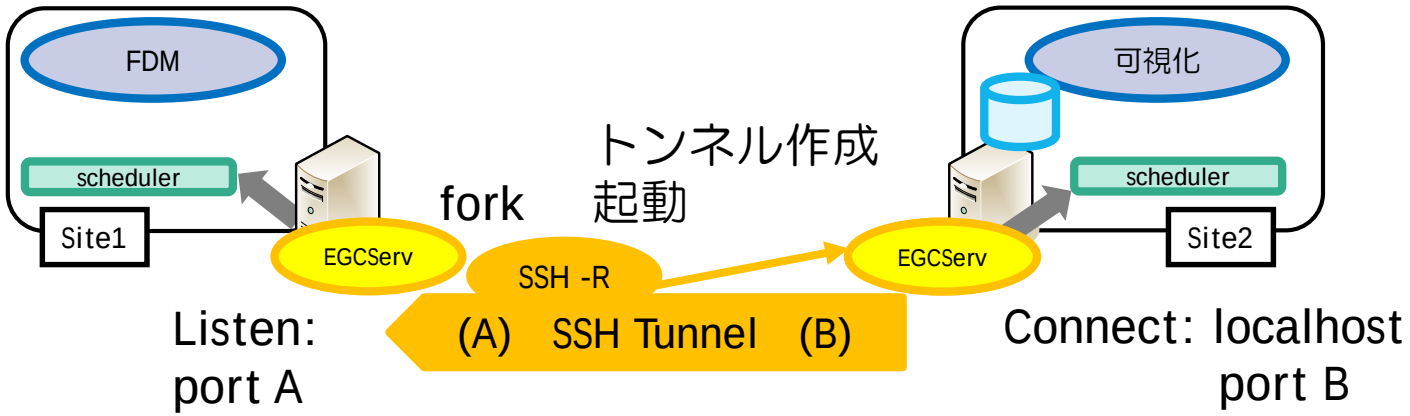
環境構築

- 1サイトからの実行による環境構築
- SSH トンネリングを利用したサーバ間転送

- 今後の再起動の為に、ユーザ名とパスワードはサーバのメモリ上に保管される

環境構築

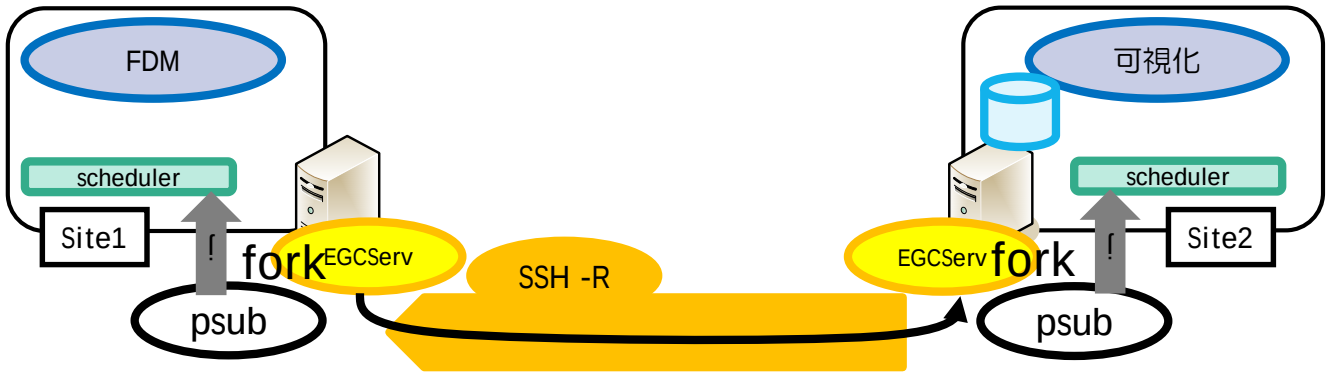
- 1サイトからの実行による環境構築
- SSH トンネリングを利用したサーバ間転送



- SSH トンネリングを行いつつ、外部サイトにサーバ起動

環境構築

- 1サイトからの実行による環境構築
- SSH トンネリングを利用したサーバ間転送



- SSHトンネル内 TCP/IP 通信によりデータ転送開始
- 指定されたプログラムを実行 (psub 等)

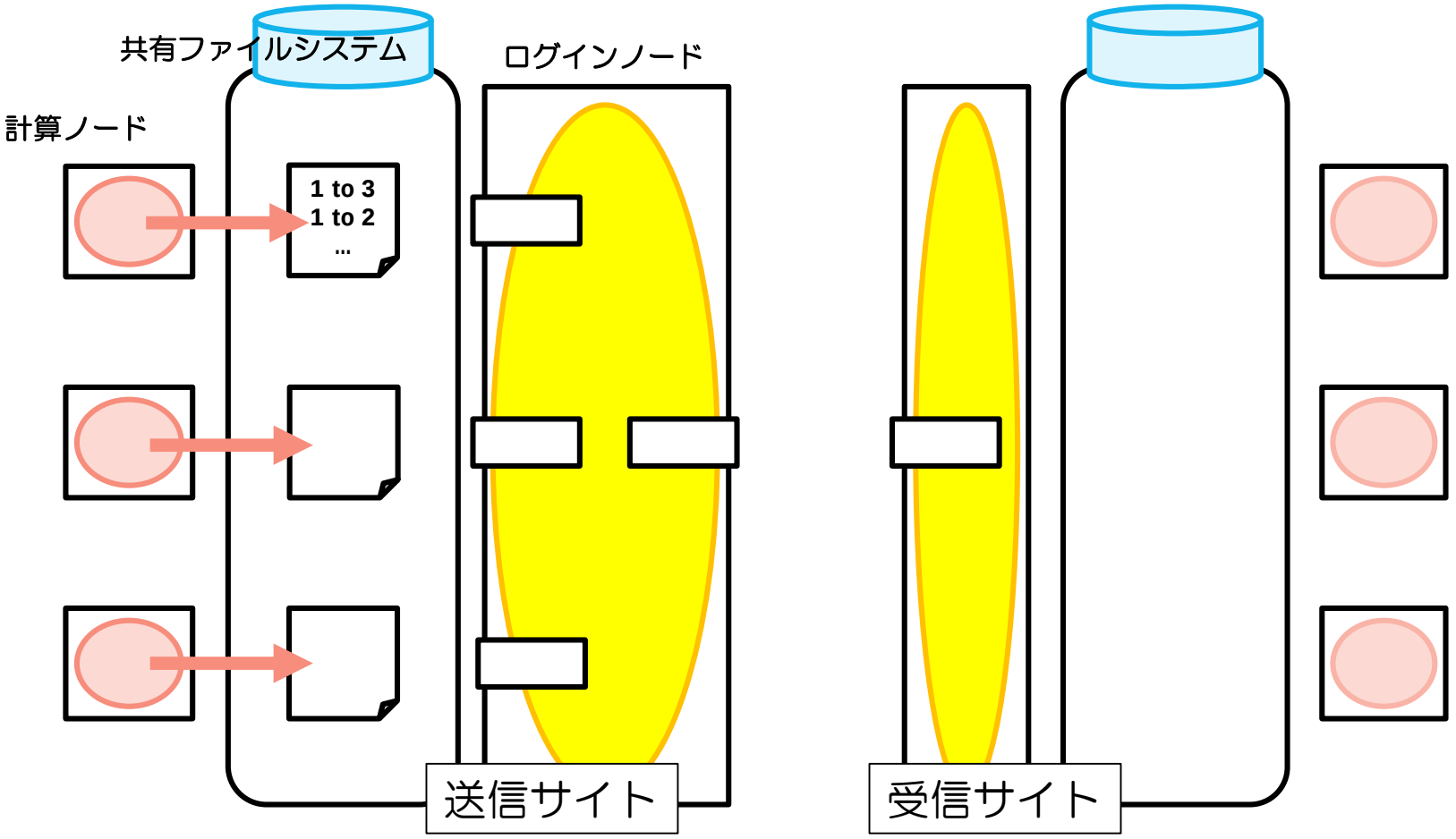
メッセージ転送

- 拠点ポリシによる通信手法の調整
  - 計算プロセスと EGCサーバ間の通信手法
    - 拠点毎に差異があるためモジュール化し拡張性を向上 (SSH/ストレージ)
- 転送先 EGC サーバにおいてメッセージをバッファリング
  - 拠点間通信が発生した際に同時に連成アプリケーションが起動しているとは限らない
  - 連成相手のアプリは実行時に自拠点のEGCサーバに問い合わせ、メッセージを受け取る

一方向通信である限り同期を排除可能

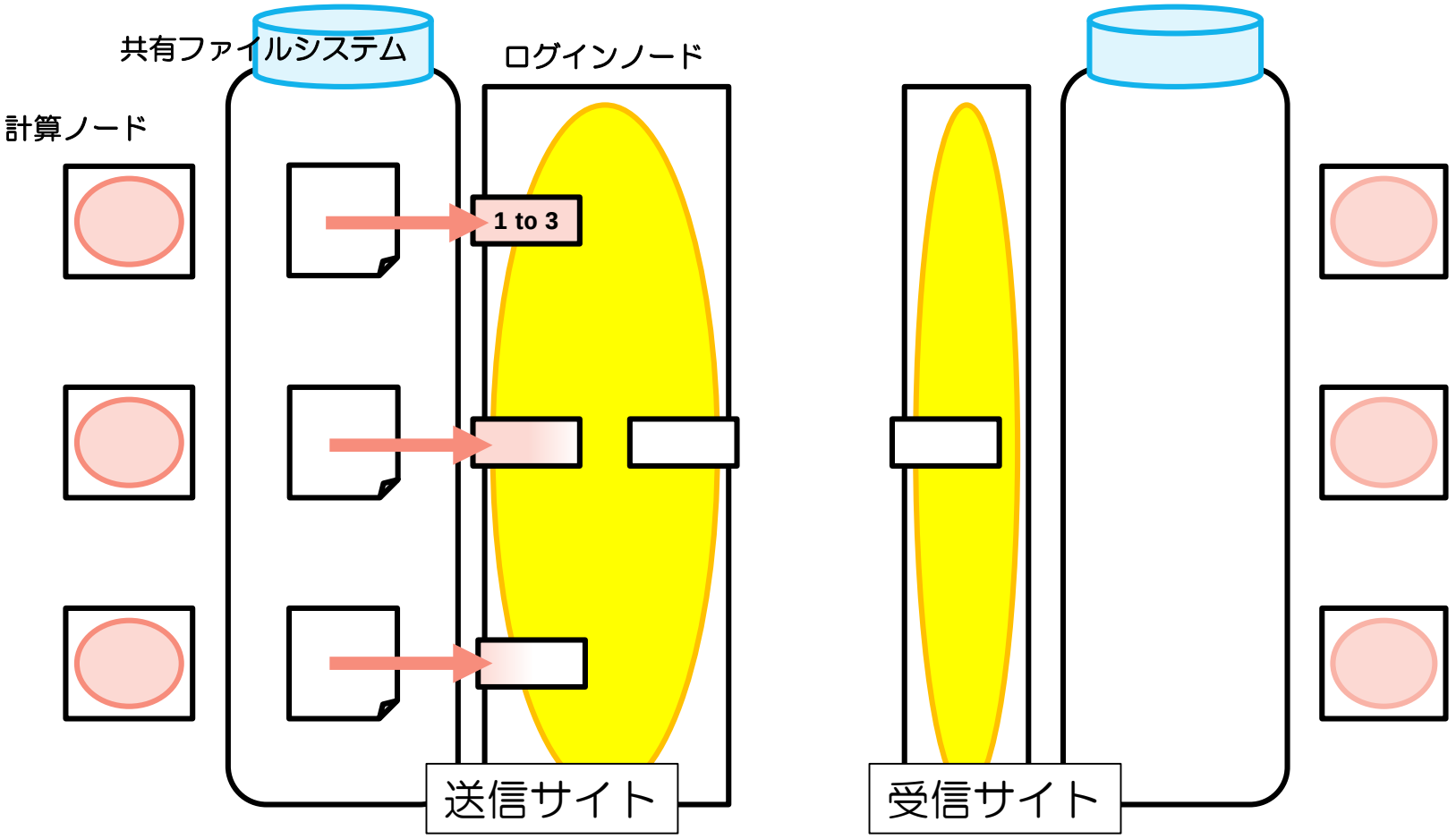
メッセージ転送 (共有ファイルシステム)

- 計算ノード上の各プロセスが通信をファイルとして書き出す。
- メッセージはパケットサイズに分割され、ヘッダをつけられる



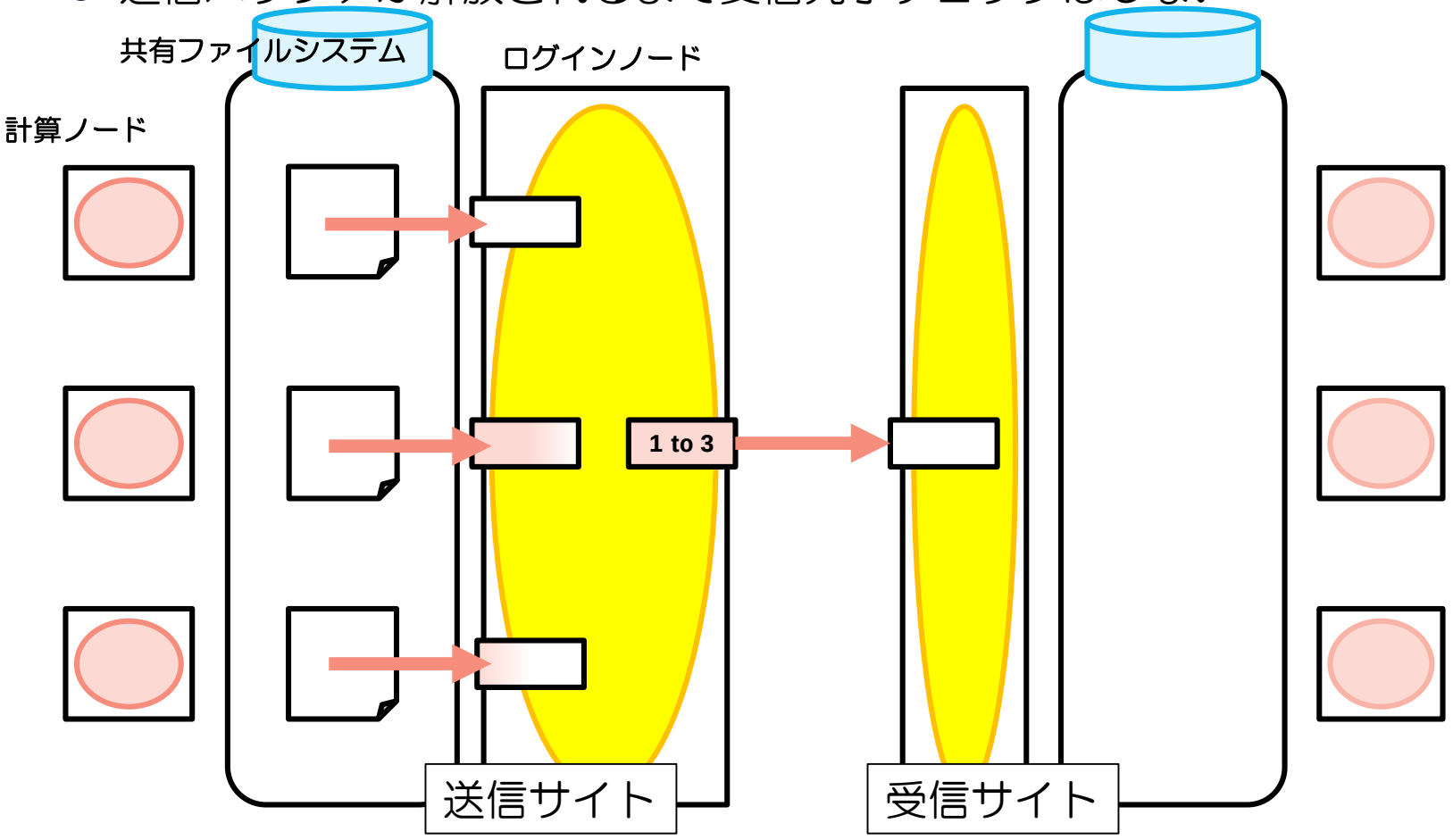
メッセージ転送 (共有ファイルシステム)

- EGCサーバは計算ノードからの出力を非同期受信 (AIO)
- EGCサーバの受信は libevent によるイベントドリブンで管理



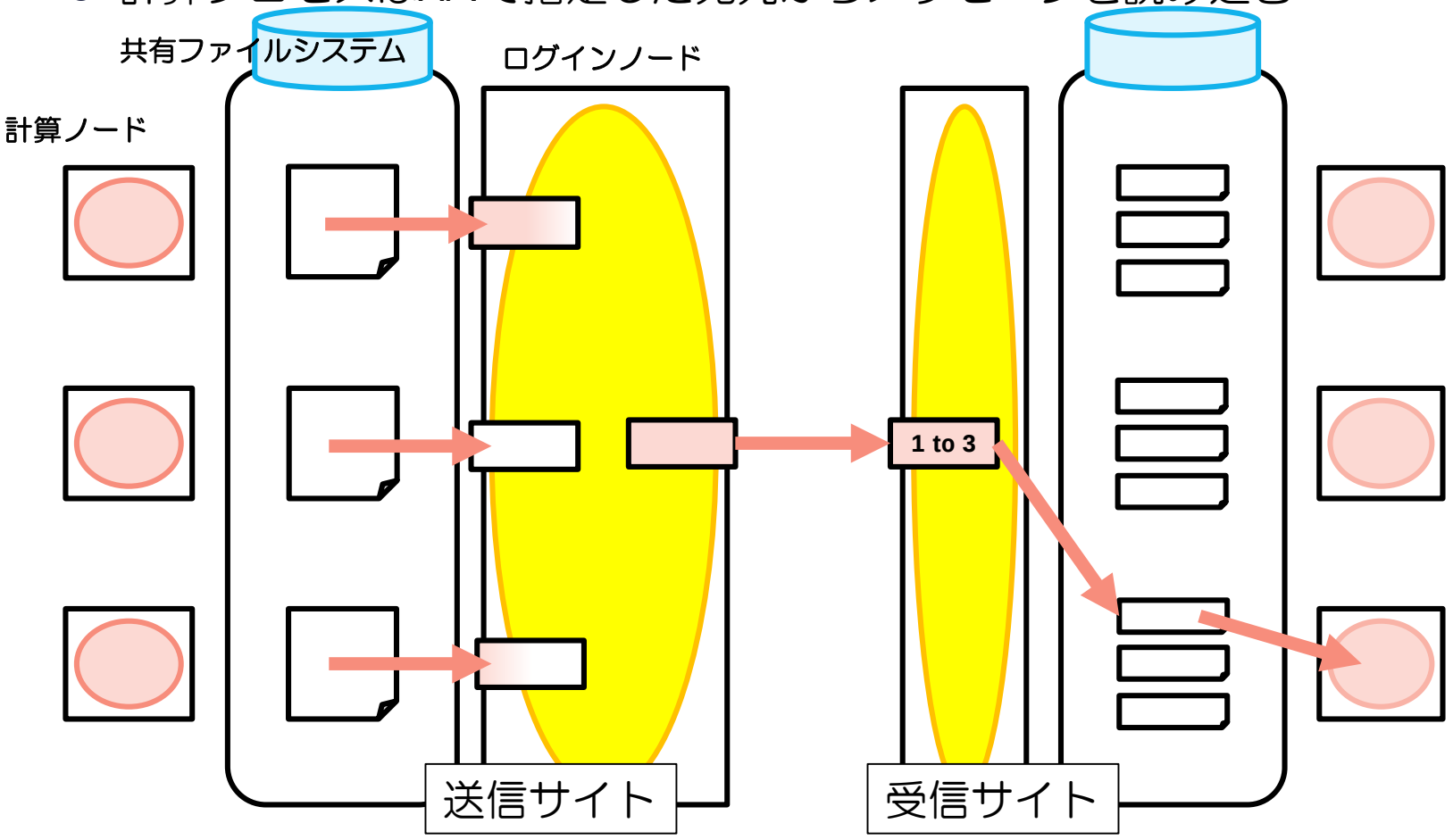
メッセージ転送 (共有ファイルシステム)

- パケットサイズ分読み込んだ受信バッファは送信バッファと交換され、再度非同期通信を開始
- 送信バッファが解放されるまで受信完了チェックはしない



メッセージ転送 (共有ファイルシステム)

- 受信側のEGCサーバは、送信元と送信先のペア毎に通信用のファイルを作成・書き込み
- 計算プロセスはAPIで指定した宛先からメッセージを読み込む



利用シナリオ

- 拠点間通信API により、プログラムを改造
- EGC Serv 用の設定ファイルの記述
  - ジョブの依存関係の記述(1方向に制限)
  - 各拠点の定義、アドレスや使われるファイルのローカルな位置、作業ディレクトリの指定
- 各拠点で利用するジョブ投入スクリプトの作成
  - 設定ファイルと一致したジョブIDと設定ファイルの位置を環境変数で追加
- 依存関係のルートジョブを行う拠点での EGC Serv の起動
- EGC Serv の外部起動に基づく SSH 認証

```
Int main(){
  EGC_init();
  for(i){
    calculation();
    communication();

    if(i % x == 0){
      EGC_Send(s, d, m);
      // or EGC_Recv(s, d, m);
    }
  }
}
```

```
#!/bin/sh
# nhost XX
# queue XX

export EGC_JID %d
export EGC_Config %s

./program args
```

インストール

- 必須環境
  - Libevent2
    - パッケージ提供のものはlibevent1 かもしれないので注意
    - ソースコードは configure –-prefix=XX; make; make install でOK。LD_LIBRARY_PATHなどの設定を忘れない
  - GCC
    - gccでしか利用検証していないので。
    - 独自仕様はデバッグ用にしか使ってないので、大丈夫かもしれない
    - egc_sys.h に #define DEBUG_ENABLED が書いてあったら消せば使われなくなる

EGCPOPS_X.X.TAR.GZ

- 解凍

  $ tar egcpops_x.x.tar.gz
  $ cd egcpops

- Makefile の編集

  $ vi Makefile
  LIBEVENT_DIR = /home/jitumoto/opt/levent
  ここをlibevent のインストール位置に変更

  $ cd sample; vi Makefile

  LIBEVENT_DIR = /home/jitumoto/opt/levent

  同様にlibevent のインストール位置に変更

- make

  $ make allclean; make; make samples

  完成するファイルは egcpops

ディストリビュート

- 利用する拠点すべてで、EGCPOPSの配備が必要
  - 当たり前だが、アーキテクチャが違う場合が多いのでバイナリのコピーではダメ

CONFIGファイル

- 解凍直下ディレクトリに spec_config.txt がありそれに書いてある・・・がめんどくさいので sample/config を編集する
  - EGC_HOSTNAME0, 1
    - ログインノードのホスト名、SSH に使われるので使えるものを
  - EGC_JOBSUBMIT0, 1
    - ログインノードで実行されるプログラム、多くの場合は psub sub.sh とかにする。
    - テストでは直接ログインノードでアプリを実行してしまっているので sample/ca.sh などが指定されている
  - EGC_INTRA_HOST_LISTENPATH0, 1
    - egcpops とやり取りする通信用ファイルその１
    - とりあえずログインノードから見れる場所を指定（共有ストレージ）
  - EGC_INTRA_HOST_DATAPATH0, 1
    - egcpops とやり取りする通信用ファイルその２
    - 通信のキャッシングファイル名になる
    - とりあえずログインノードから見れる場所を指定（共有ストレージ）
  - EGC_INTRA_MODE0, 1
    - file 固定で
  - EGC_POPSERV_PATH0, 1
    - egcpops がおいてある場所、フルパス指定が無難
  - EGC_CONFIG0, 1
    - 本コンフィグファイルがおいてある場所、フルパス指定が無難

JOB投入ファイル、実行ファイル

- EGC_JOBSUBMIT%d に指定するファイル
- シェルスクリプトにするべき
  - 環境変数で JOBID と config の位置だけ設定する必要がある。
    （どうにも内部的に引継ぎできなかった
  - Config ファイルに合わせて
    - EGC_JID：ジョブID
    - EGC_CONFIG：コンフィグファイルの場所（フルパスが無難）
- 実行ディレクトリ
  - egcpops を起動させるノード(JOBID=0)では、egcpops を起動させた場所
  - 他のリモートノードではホームになる。

→適宜シェルスクリプト内で移動をかける

実行方法 (注意あり)

- $ egcpops –c config
  - -c を付けない場合は同じディレクトリ内の egc.conf を開きに行く

- 注意
  - 実行終了後のegcpops 終了部分の実装ができてない
    - EGC_JOBSUBMITで指定したプログラムの終了を検知しないので、egcpops を cntl+Cで終わらせる必要あり
    - egcpops をEGC_JOBSUBMIT プログラム終了前に中断した場合 EGC_JOBSUBMIT プログラムが残る killall する。
    - 他端末で入って環境ごとに決まったjob 状態確認コマンドを使ってジョブ終了を確認ください（すみません
  - 実行後、中間ファイルが消えないので消す必要がある
    - EGC_INTRA_LISTENPATH, EGC_INTRA_DATAPATH
    - ただしコンシューマ(受信側)はキャッシュとして機能するので、残してある場合、コンシューマジョブ単体を投入すれば動く

SSH 認証

- 一度 SSH コマンドで接続して known_hosts などを集めておくこと
- ssh-agent との同時利用は試してない…たぶん動かない

- 起動後
  [Concertino1] User Name:
  [Concertino1] Password or Pass phrase:
  とホストごとに聞かれるので入力、なお password は入力内容が表示されない。

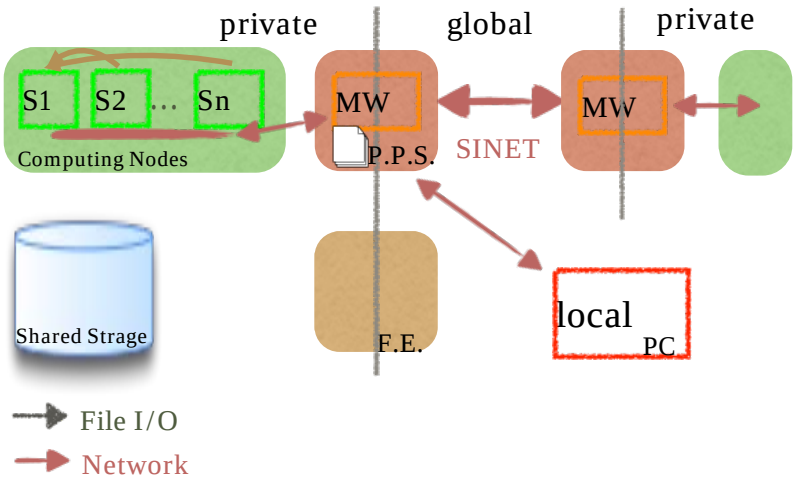
サンプルについて (SAMPLE/以下)

注意：config ファイルのパス系は書き換え必須

- ローカルで ClientA, リモートでClientB を実行する
  - ClientA: リモートに 0-1023のint 配列1024個を2回送る
  - ClientB: リモートから int 配列1024個を2回受け取り、それぞれ表示する
- ca.sh, cb.sh が起動される
  - 出力はそれぞれ XX.out XX.err に
  - XX.out に 0-1023 が2回出力されれば正常

今後の展望

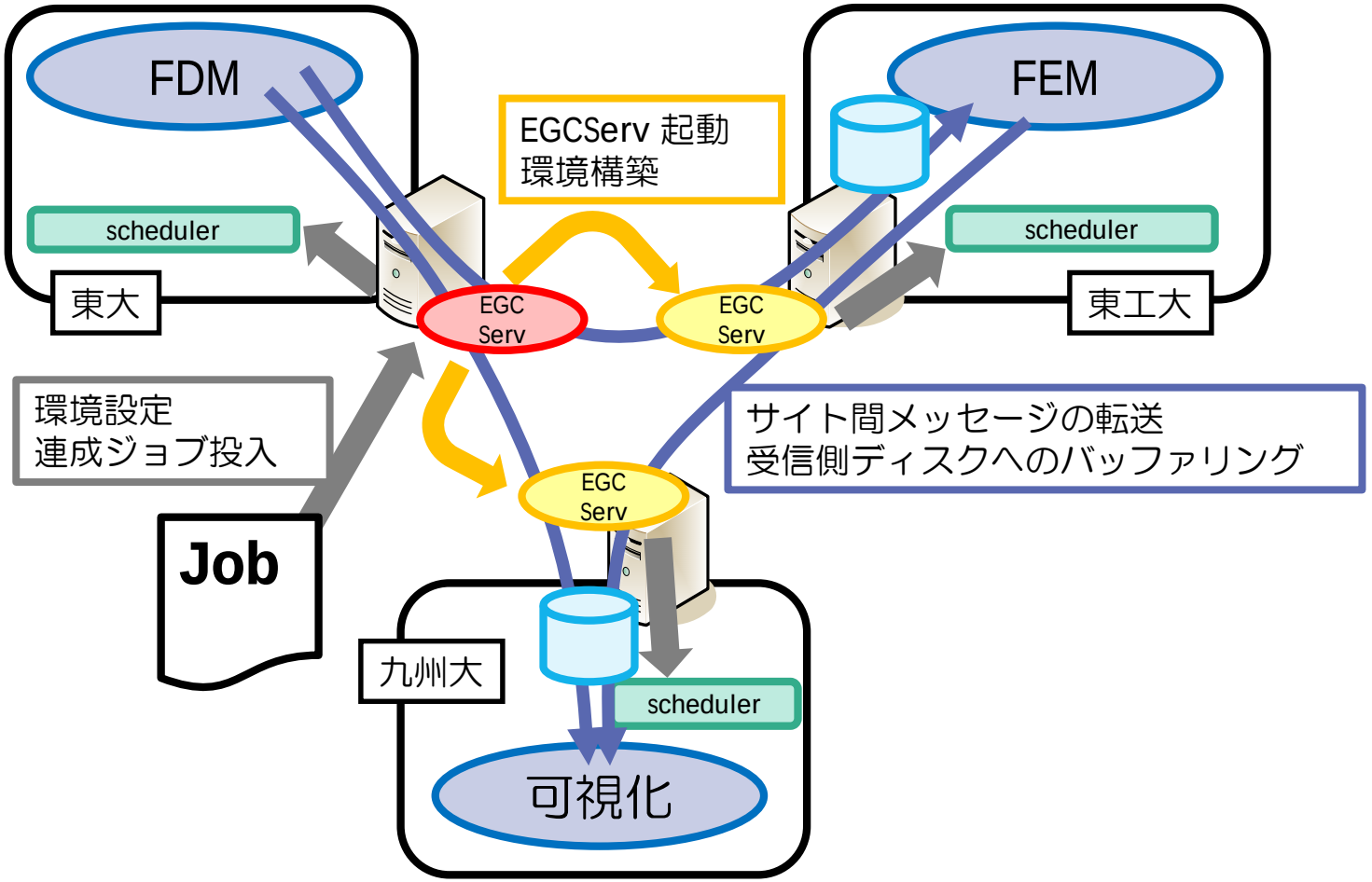
- フレームワークを利用したアプリケーションの大規模評価
  - 地震波動ー建築物振動連成解析と可視化
  - 転送量・頻度によるスループットの評価
    - 利用シナリオ的には本質的ではない
- OpenFORM との連携
  - 九大チームの行っているOpenFORM連成ツールに提案フレームワークでの大域化を適用



今後の展望

- アプリケーション評価からフィードバックした機能拡張
  - サイト間メッセージの自動最適化
    - ブロックサイズ
    - メッセージ量（頻度）の自動調整
  - ３拠点以上での連成アプリケーション
- クラウド環境との協調検討
- 既存アプリケーションの改変をより少なくする手法の検討
  - MPI のラップ: ユーザから与えられるRANK配置を元に自動的に拠点間通信と拠点内通信を判別
- 多種環境でつかってもらえるよう安定化・公開を行う
  - HPCI だとサイト間通信を gfarm で置き換えられる(?)

3拠点連携


FDM
FEM
EGCServ 起動
環境構築
scheduler
scheduler
東大
東工大
EGC
Serv
EGC
Serv
環境設定
連成ジョブ投入
サイト間メッセージの転送
受信側ディスクへのバッファリング
EGC
Serv
Job
九州大
scheduler
可視化