

# MADOCA II データ収集と 蓄積システム

山下明広 籠正裕 酒井久伸

JASRI/SPring-8

2013/8/5 第10回日本加速器学会年会

# MADOCA

- **Message And Database Oriented Control Architecture**

# MADOCA

- Message And **Database** Oriented Control Architecture

# MADOCA

- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用

# MADOCA

- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用
  - HiSOR, NewSUBARU, SCSS, SACLA

# MADOCA

- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用
  - HiSOR, NewSUBARU, SCSS, SACLA
- データベース2つの用途

# MADOCA

- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用
  - HiSOR, NewSUBARU, SCSS, SACLA
- データベース2つの用途
  1. パラメーター管理

# MADOCA

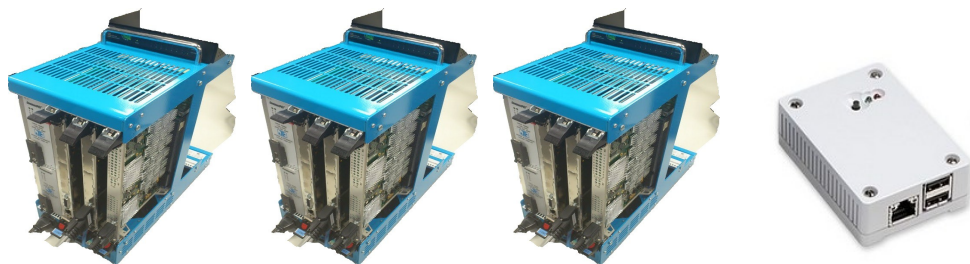
- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用
  - HiSOR, NewSUBARU, SCSS, SACLA
- データベース2つの用途
  1. パラメーター管理
  2. ログデータ蓄積



# MADOCA

- Message And **Database** Oriented Control Architecture
- 1997年のSPring-8コミッショニング以来使用
  - HiSOR, NewSUBARU, SCSS, SACLA
- データベース2つの用途
  1. パラメーター管理
  2. **ログデータ**蓄積

# 現MADOCA ログデーター収集系+蓄積

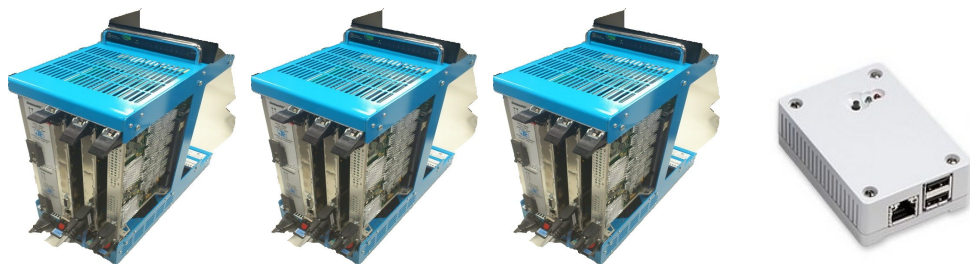


~500組み込みコンピュータ

# 現MADOCA ログデーター収集系+蓄積

Poller/collector  
Process

Poller/collector  
Process

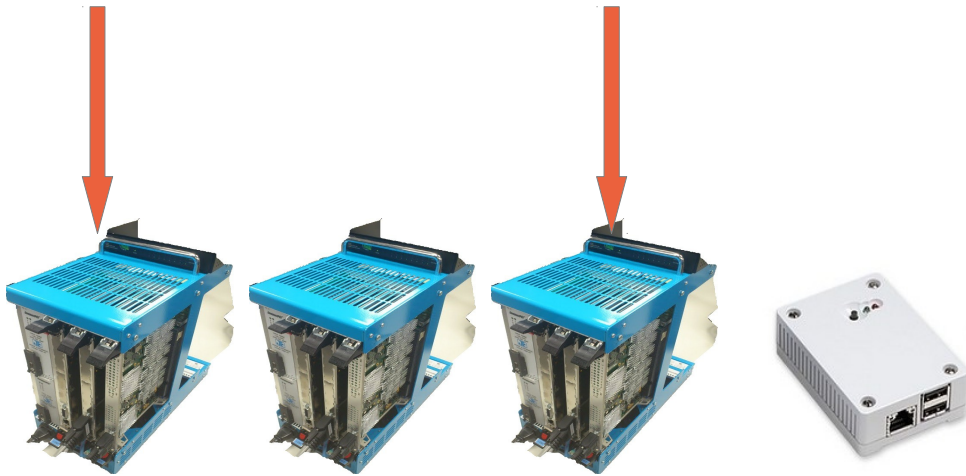


~500組み込みコンピュータ

# 現MADDOCA ログデータ収集系+蓄積

Poller/collector  
Process

Poller/collector  
Process

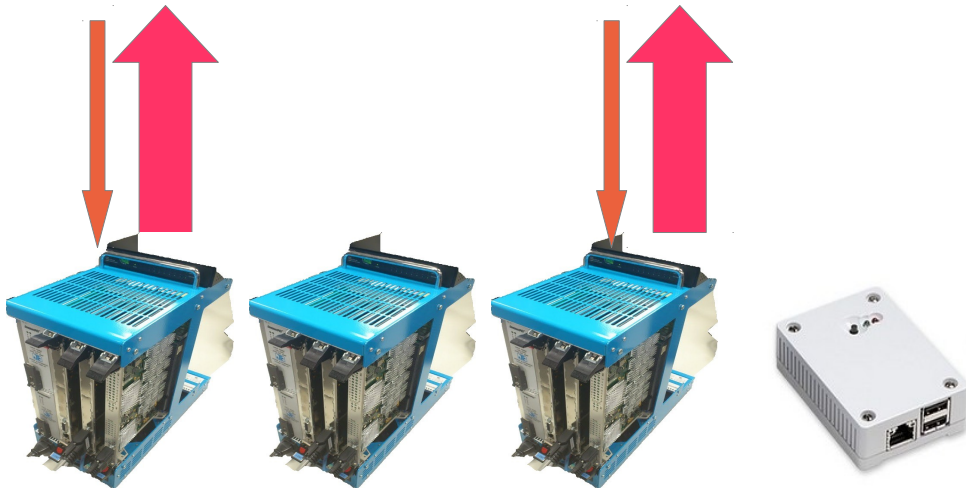


Query:問い合わせ  
“データーください”

# 現MADDOCA ログデータ収集系+蓄積

Poller/collector  
Process

Poller/collector  
Process



Reply: 返事  
データの集合

# 現MADOCA ログデータ収集系+蓄積

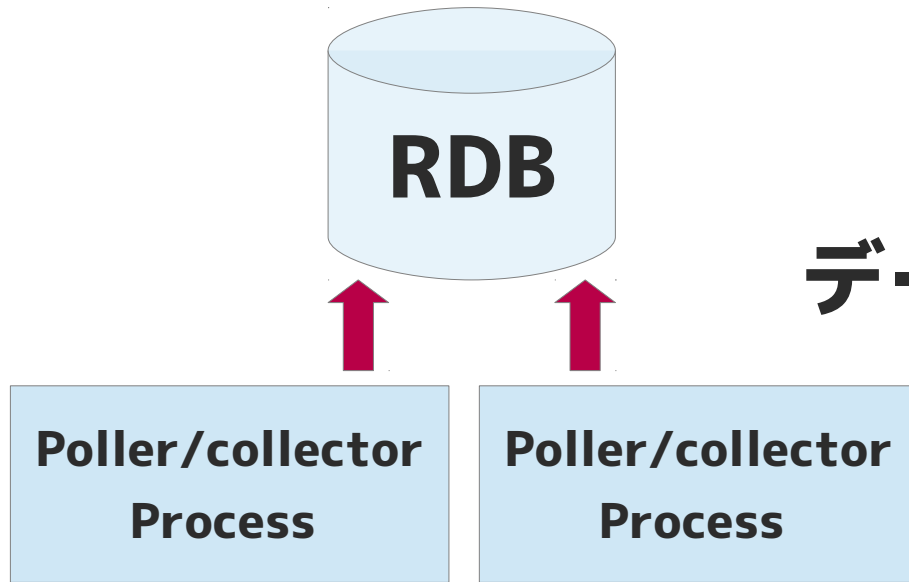
Poller/collector  
Process

Poller/collector  
Process

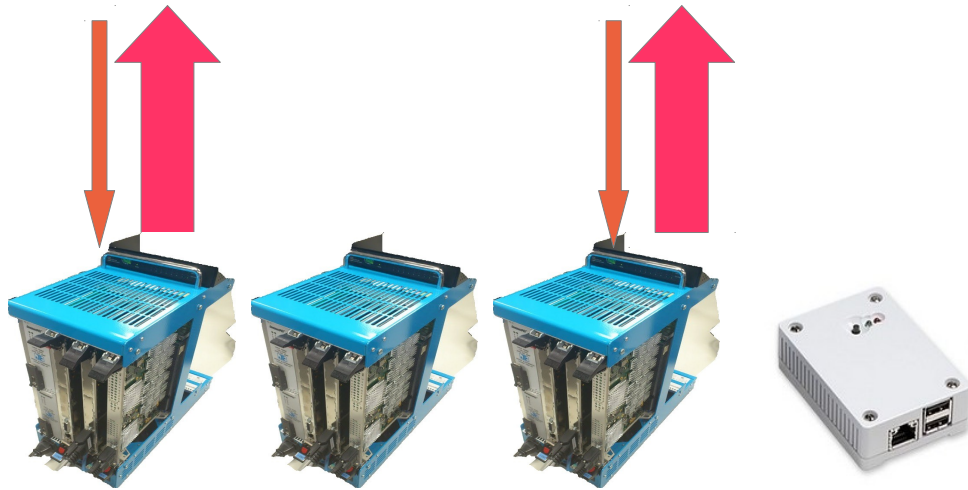
RPCによる定期的なQuery/Reply



# 現MADDOCA ログデータ収集系+蓄積



データベースへの書き込み



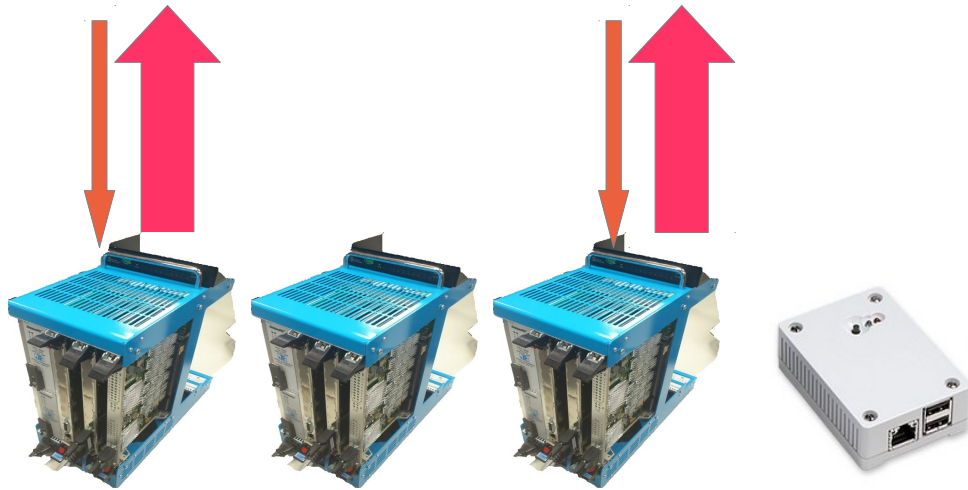
# 現MADDOCA ログデータ収集系+蓄積

73 SQLs/sec  
(7800 signals/sec)



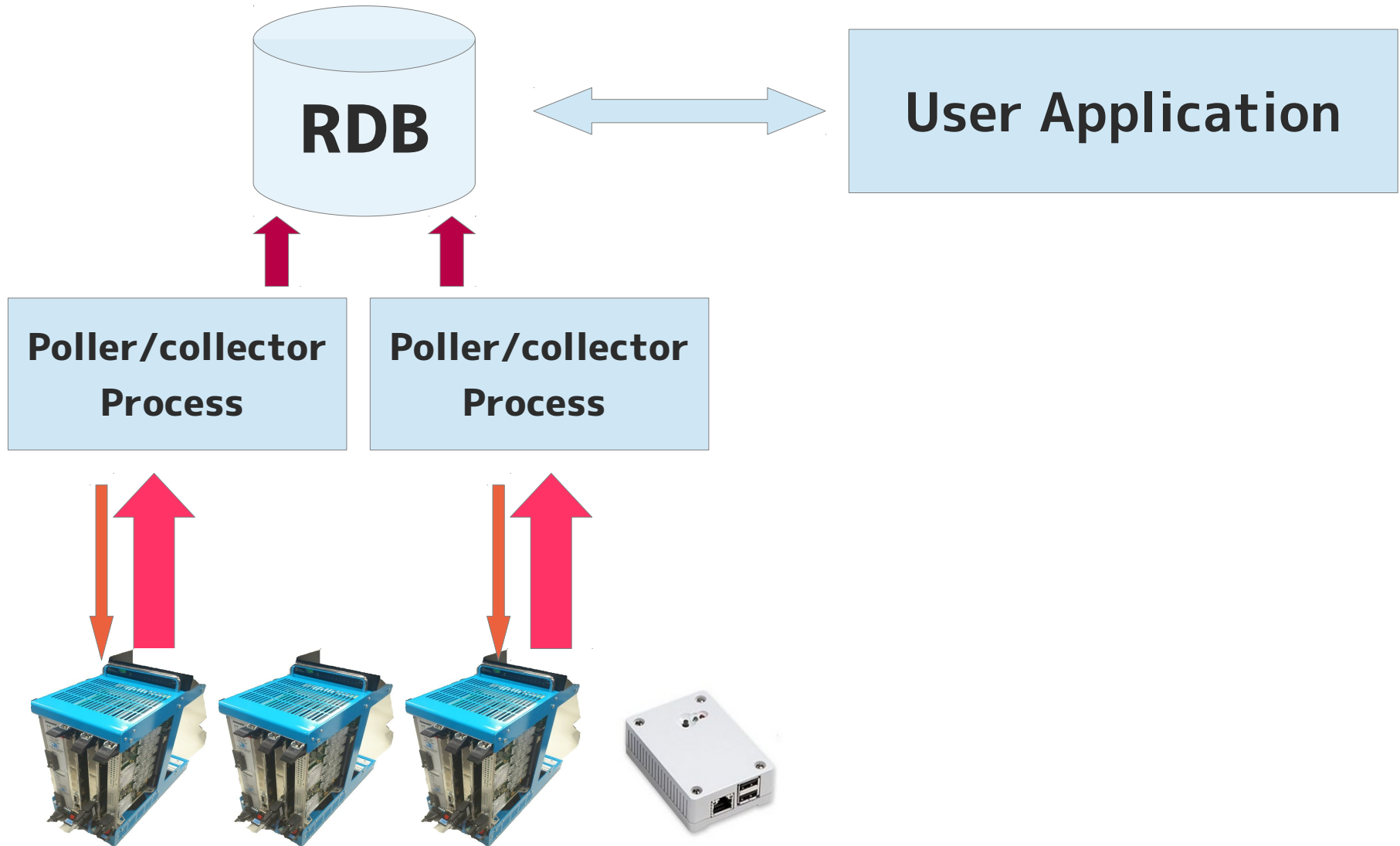
Poller/collector  
Process

Poller/collector  
Process

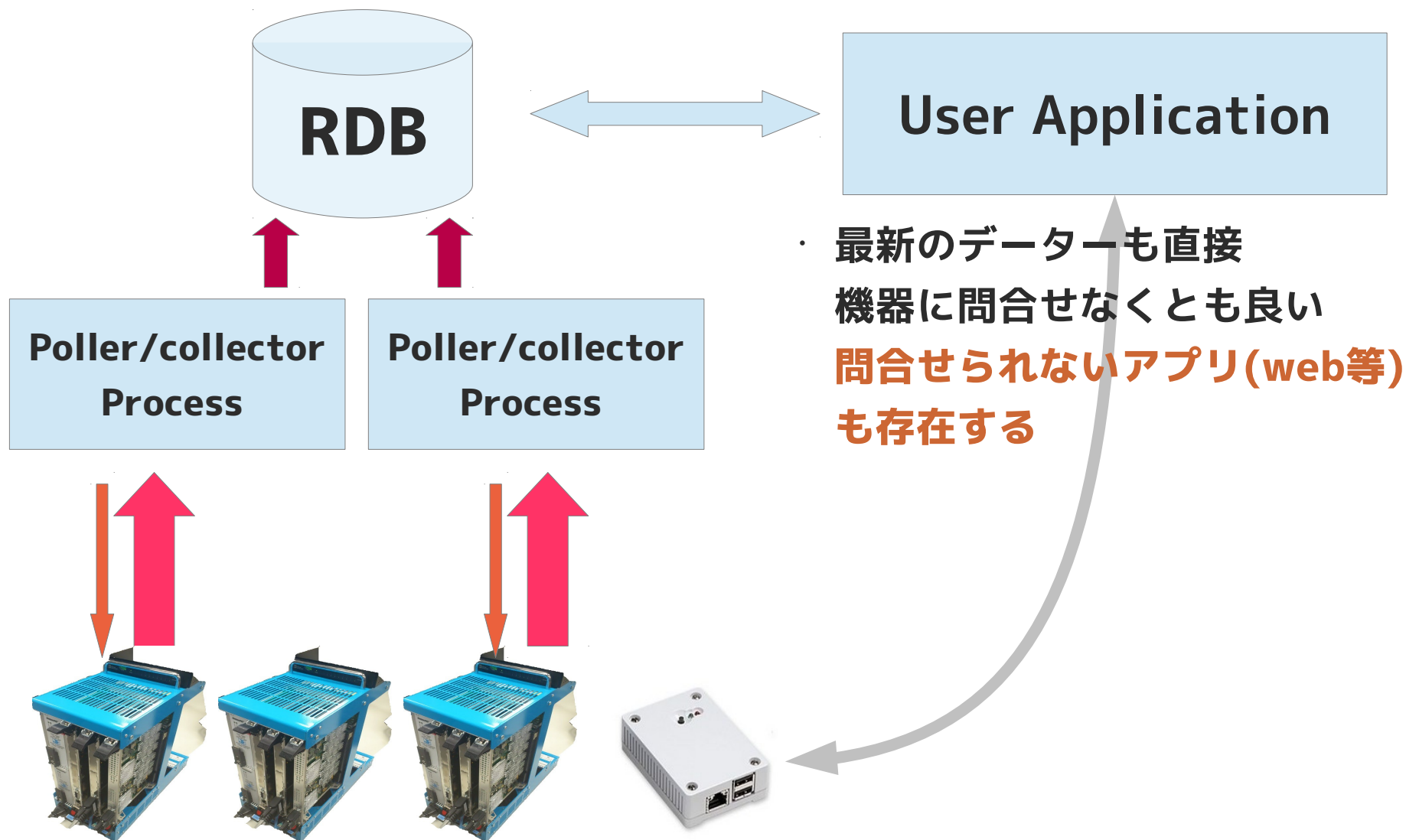




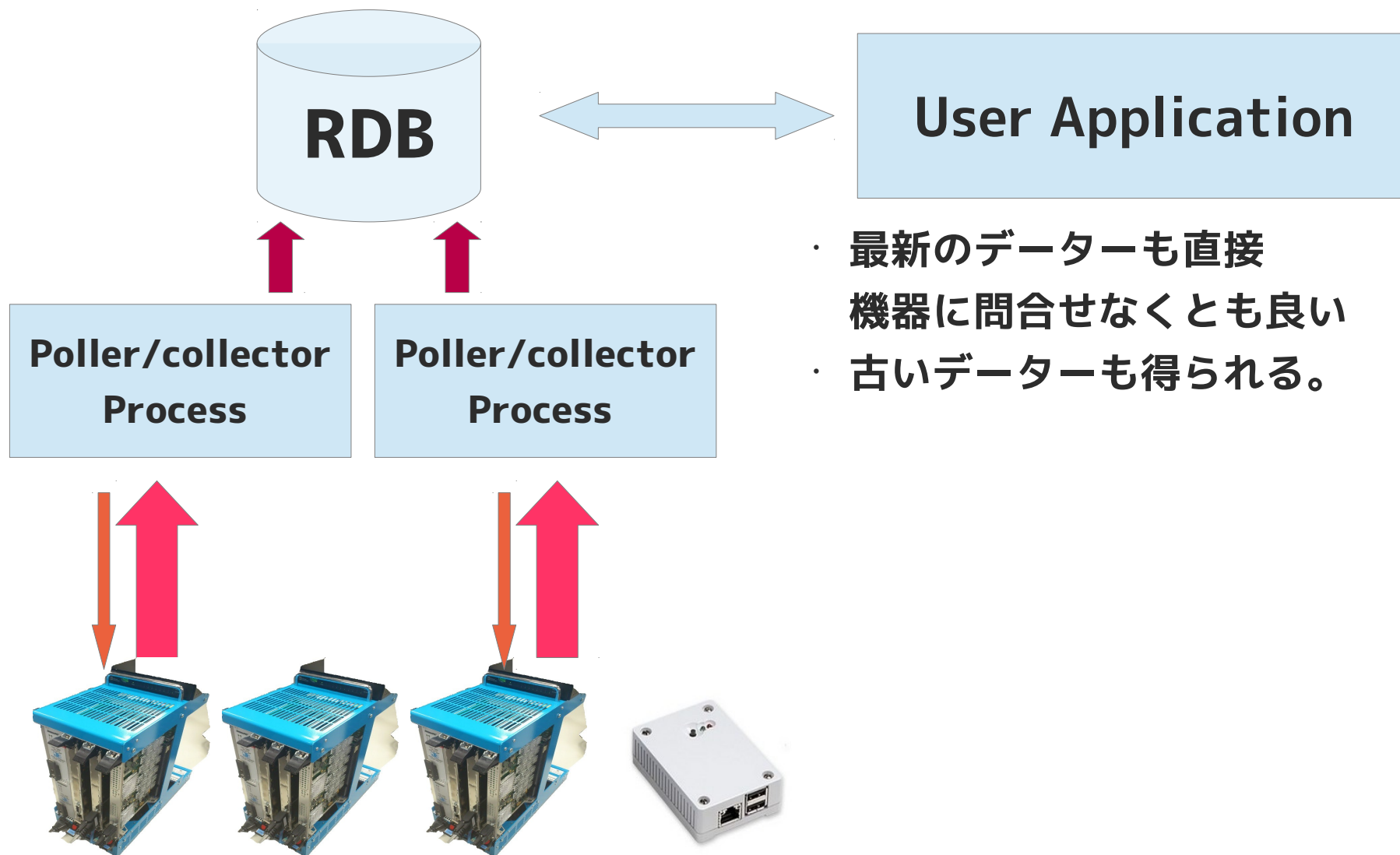
# 現MADDOCA ログデータ収集系+蓄積



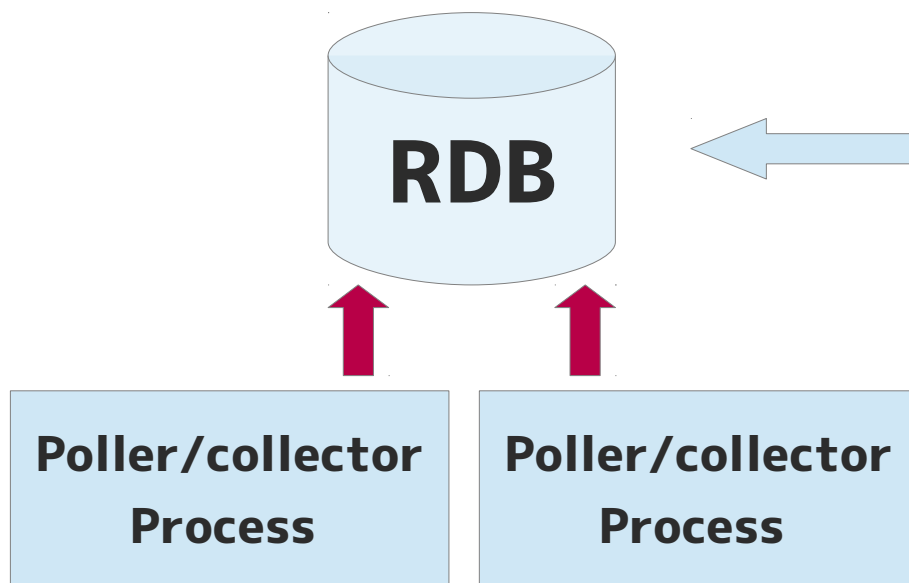
# 現MADDOCA ログデータ収集系+蓄積



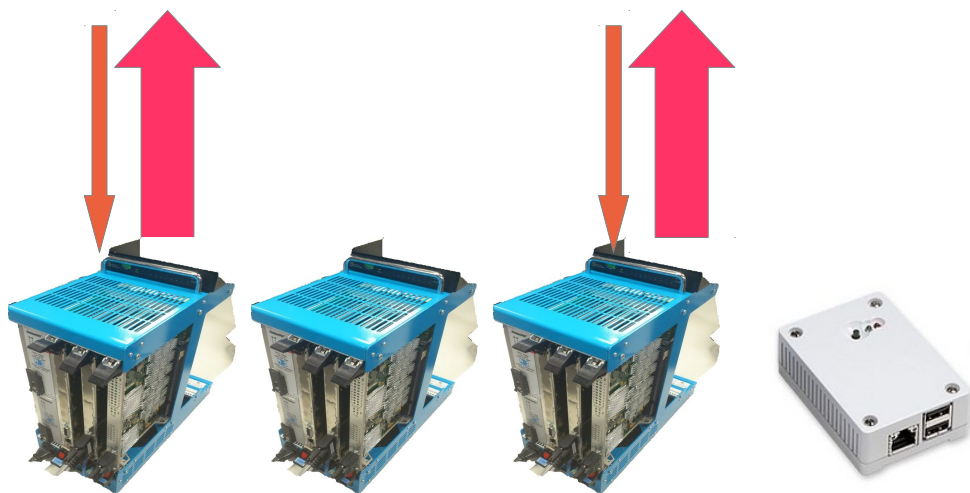
# 現MADDOCA ログデータ収集系+蓄積



# 現MADDOCA ログデータ収集系+蓄積



- ・ 最新のデータも直接機器に問合せなくとも良い
- ・ 古いデータも得られる。
- ・ ヒューマンリーダブルな信号名(SVOC)で問い合わせる。



# MADDOCA ログデータベース

# MADDOCA ログデータベース

## 成功の3つの鍵

# MADDOCA ログデータベース

## 1. 時間的完全性

全ての時間のデータを蓄積してアクセス可能  
たとえ1997年のデータでも

# MADDOCA ログデータベース

**1. 時間的完全性**

**2. 空間的完全性**

**全ての機器のデータにアクセス可能**



# MADOCA ログデータベース

**1. 時間的完全性**

**2. 空間的完全性**

**3. 均一性**

**全てのデーターに同じ方法でアクセスできる**

**電磁石、真空、LINAC、ビームライン、、、**

**S/V/O/C 文法**

# MADDOCAログデータベース=RDB

- リレーショナルデータベース

# MADDOCAログデータベース=RDB

- リレーショナルデータベース
  - 全てを2次元のテーブルで表現

time0	value0	value1	....	valueN
time1	value0	value1	....	valueN
time2	value0	value1	....	valueN
time3	value0	value1	....	valueN

# MADDOCAログデータベース=RDB

- リレーショナルデータベース
  - 全てを2次元のテーブルで表現

time0	value0	value1	....	valueN
-------	--------	--------	------	--------

# MADDOCAログデータベース=RDB

- リレーショナルデータベース
  - **全てを2次元のテーブルで表現**

time0	value0	value1	....	valueN
time1	value0	value1	....	valueN

# MADDOCAログデータベース=RDB

- リレーショナルデータベース
  - **全てを2次元のテーブルで表現**

time0	value0	value1	....	valueN
time1	value0	value1	....	valueN
time2	value0	value1	....	valueN

# MADDOCAログデータベース=RDB

- リレーショナルデータベース
  - 全てを2次元のテーブルで表現

time0	value0	value1	....	valueN
time1	value0	value1	....	valueN
time2	value0	value1	....	valueN
time3	value0	value1	....	valueN

# MADDOCAログデータベース=RDB

- **リレーショナルデータベース**
  - **全てを2次元のテーブルで表現**
  - **当時それが主流になりかけていた**



# MADDOCAログデータベース=RDB

- **リレーショナルデータベース**
  - **全てを2次元のテーブルで表現**
  - **当時それが主流になりかけていた**
  - **必ずしもログデータベースに向いているわけではない**

# MADDOCAログデータベース=RDB

- **リレーショナルデータベース**
  - **全てを2次元のテーブルで表現**
  - **当時それが主流になりかけていた**
  - **必ずしもログデータベースに向いているわけではない**
  - **が3つの条件を満足させ、安定に運用してきた**

# MADDOCAログデータベース=RDB

- **リレーショナルデータベース**
  - **全てを2次元のテーブルで表現**
  - **当時それが主流になりかけていた**
  - **必ずしもログデータベースに向いているわけではない**
  - **が3つの条件を満足させ、安定に運用してきた**

# RDBをログデータに使用する

# RDBをログデータに使用する

**将来の加速器制御に  
使用するための問題点が  
わかってきた**

# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない

# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
  - 新しい信号の登録=新テーブルをつくる
    - 登録へ手間がかかる

# RDBをログデータに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
  - **新しい信号の登録=新テーブルをつくる**
  - **同じテーブルの信号は同一時刻である必要**



# RDBをログデーターに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
  - **新しい信号の登録=新テーブルをつくる**
  - **同じテーブルの信号は同一時刻である必要**
    - **突発的データーは別ルートで記録しなければならない**

# RDBをログデータに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
- **性能上の問題**

# RDBをログデータに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
- **性能上の問題**

# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
- 性能上の問題
  - 性能向上のためには Scale Up (1台のコンピュータを高性能化)



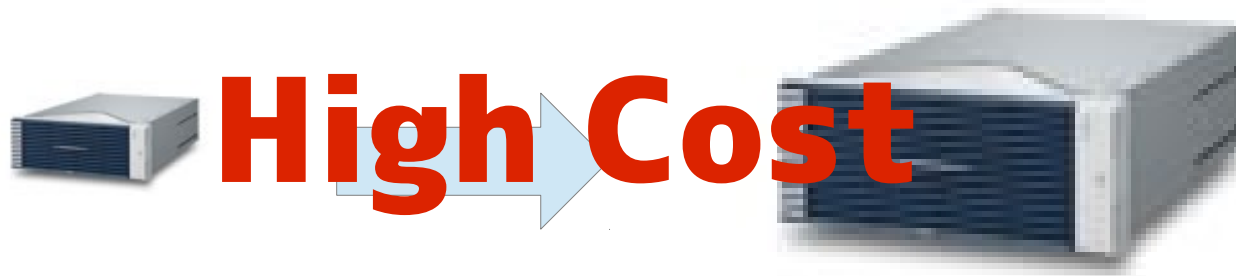
# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
- 性能上の問題
  - 性能向上のためには Scale Up (1台のコンピュータを高性能化)



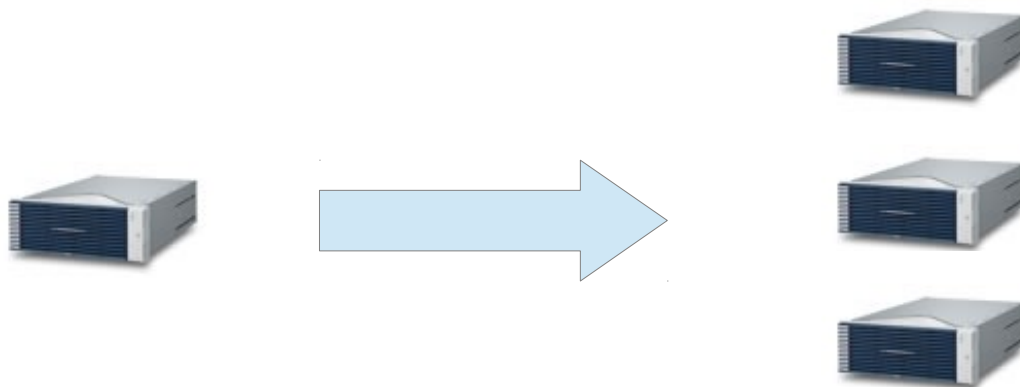
# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
- 性能上の問題
  - 性能向上のためには Scale Up (1台のコンピュータを高性能化)



# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
- 性能上の問題
  - **Scale Out (分散化)はRDBでは困難**



# RDBをログデータに使用する

- 挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない
- 性能上の問題
  - **Scale Out (分散化)はRDBでは困難**





# RDBをログデーターに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
- **性能上の問題**
- **複雑なデーター構造を単純なテーブルにマッピングするのは難しい**

# RDBをログデーターに使用する

- **挿入の性能をあげるため1テーブルに多数の信号を入れざるを得ない**
- **性能上の問題**
- **複雑なデーター構造を単純なテーブルにマッピングするのは難しい**

# MADUCA II

# MADOCA IIへ

- **MADOCAの成功の条件を満たしつつ**

# MADOCA IIへ

- **MADOCAの成功の条件を満たしつつ**
- **MADOCAの問題点を改善し**

# MADOCA IIへ

- **MADOCAの成功の条件を満たしつつ**
- **MADOCAの問題点を改善し**
- **次の20年へ**

# MODOCA II ログデーター要求

# MODOCA II ログデーター要求

- データー収集の柔軟性



# MODOCA II ログデーター要求

- データー収集の柔軟性
  - 時間的

# MODOCA II ログデーター要求

- **データー収集の柔軟性**
  - **時間的**
    - **各信号が独立したタイミングでデーター収集ができる。**

# MODOCA II ログデーター要求

- **データー収集の柔軟性**
  - **時間的**
    - **各信号が独立したタイミングでデーター収集ができる。**
  - **空間的**

# MODOCA II ログデーター要求

- **データー収集の柔軟性**
  - **時間的**
    - **各信号が独立したタイミングでデーター収集ができる。**
  - **空間的**
    - **データー追加が容易**

# MODOCA II ログデーター要求

- **データー収集の柔軟性**
  - **時間的**
    - **各信号が独立したタイミングでデーター収集ができる。**
  - **空間的**
    - **データー追加が容易**
    - **データーの型に柔軟性**

# MODOCA II ログデーター要求

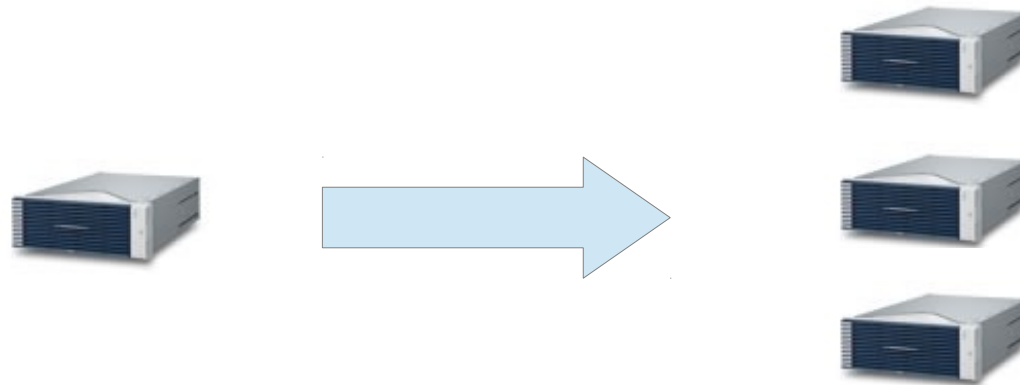
- データー収集の柔軟性
- パフォーマンス

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
  - **Scale Out型**

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
  - **Scale Out型**





# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
  - **Scale Out型**
    - データベースのみならず収集系も  
**Scale Outすること**

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
  - **Scale Out型**
    - データベースのみならず収集系も  
**Scale Outすること**
  - **現MODOCAの性能をはるかに越えること**

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
- 現在のMADOCAの財産は継承

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
- 現在のMODOCAの財産は継承
  - 信号名、 S/V/O/C文法

# MODOCA II ログデーター要求

- データー収集の柔軟性
- パフォーマンス
- 現在のMADOCAの財産は継承
  - 信号名、S/V/O/C文法
  - 成功への3つの鍵

# データ収集系

## MADOCA

Poller/collector  
Process

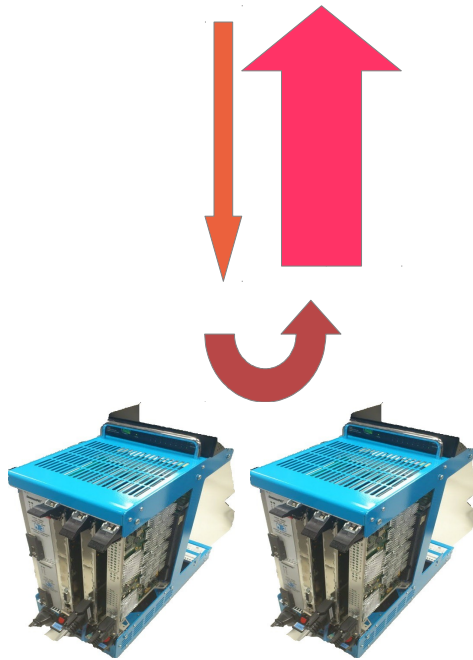
1つのqueryで多数のデータを  
いっぺんに取得 同期型



# データ収集系

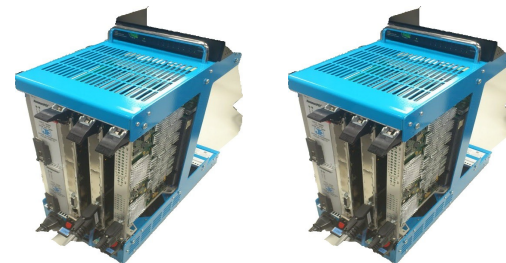
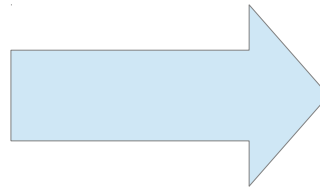
## MADOCA

Poller/collector  
Process



## MADOCA II

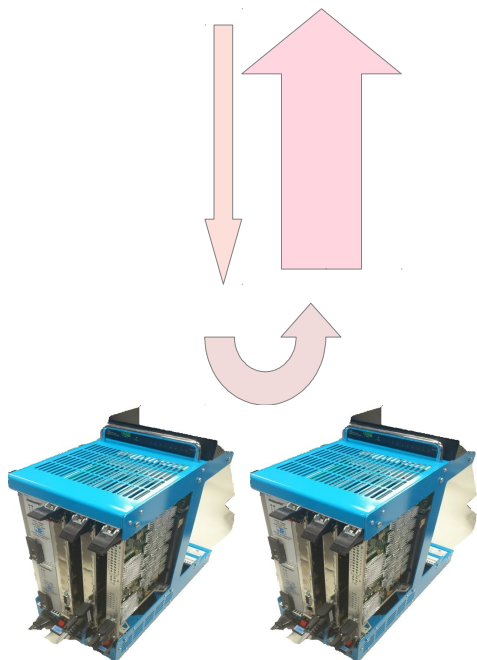
中継  
Process



# データ収集系

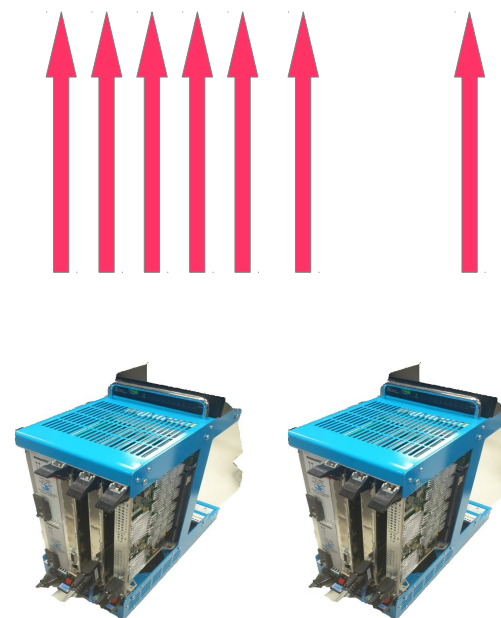
## MADOCA

Poller/collector  
Process



## MADOCA II

中継  
Process

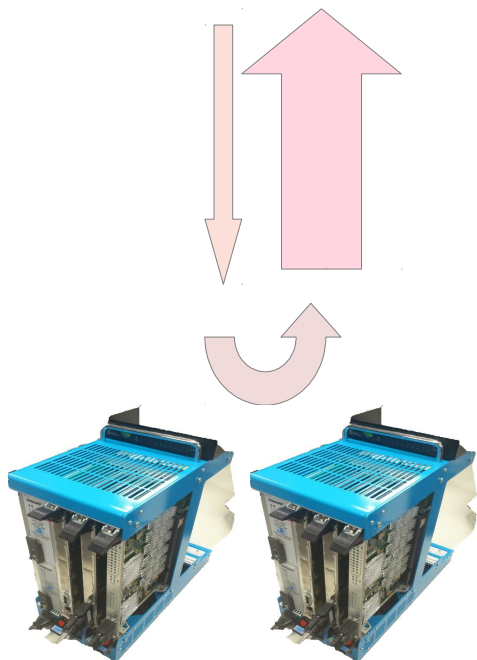




# データ収集系

## MADOCA

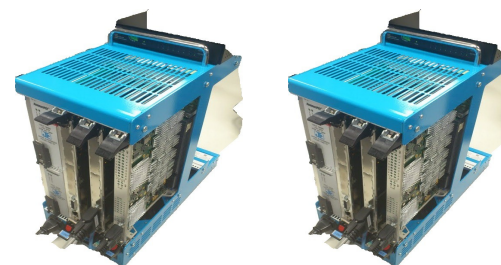
Poller/collector  
Process



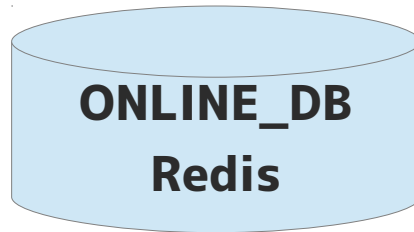
## MADOCA II

中継  
Process

一方的に多数のデータを  
個々に送る 非同期型



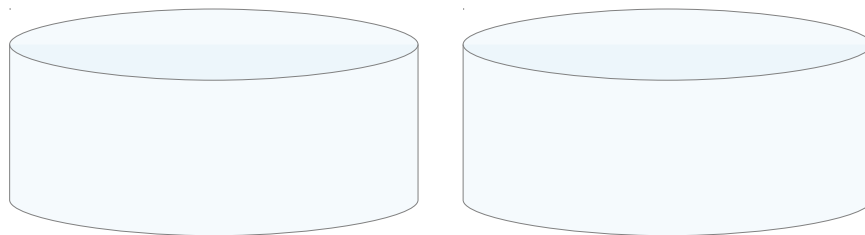
# MADOCALLログデータ収集系 実装



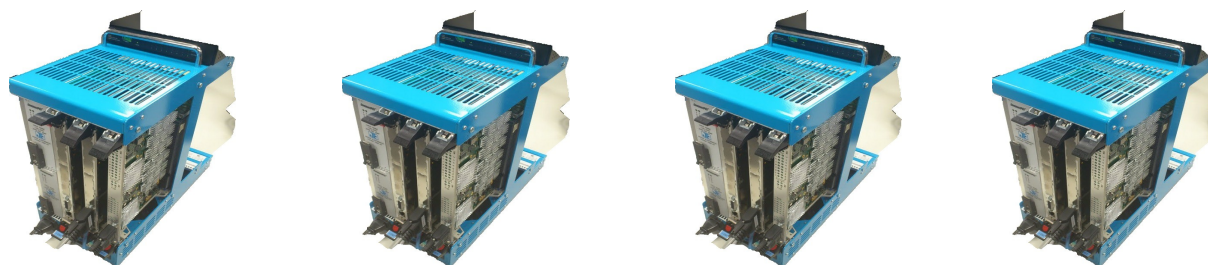
書き込みprocess



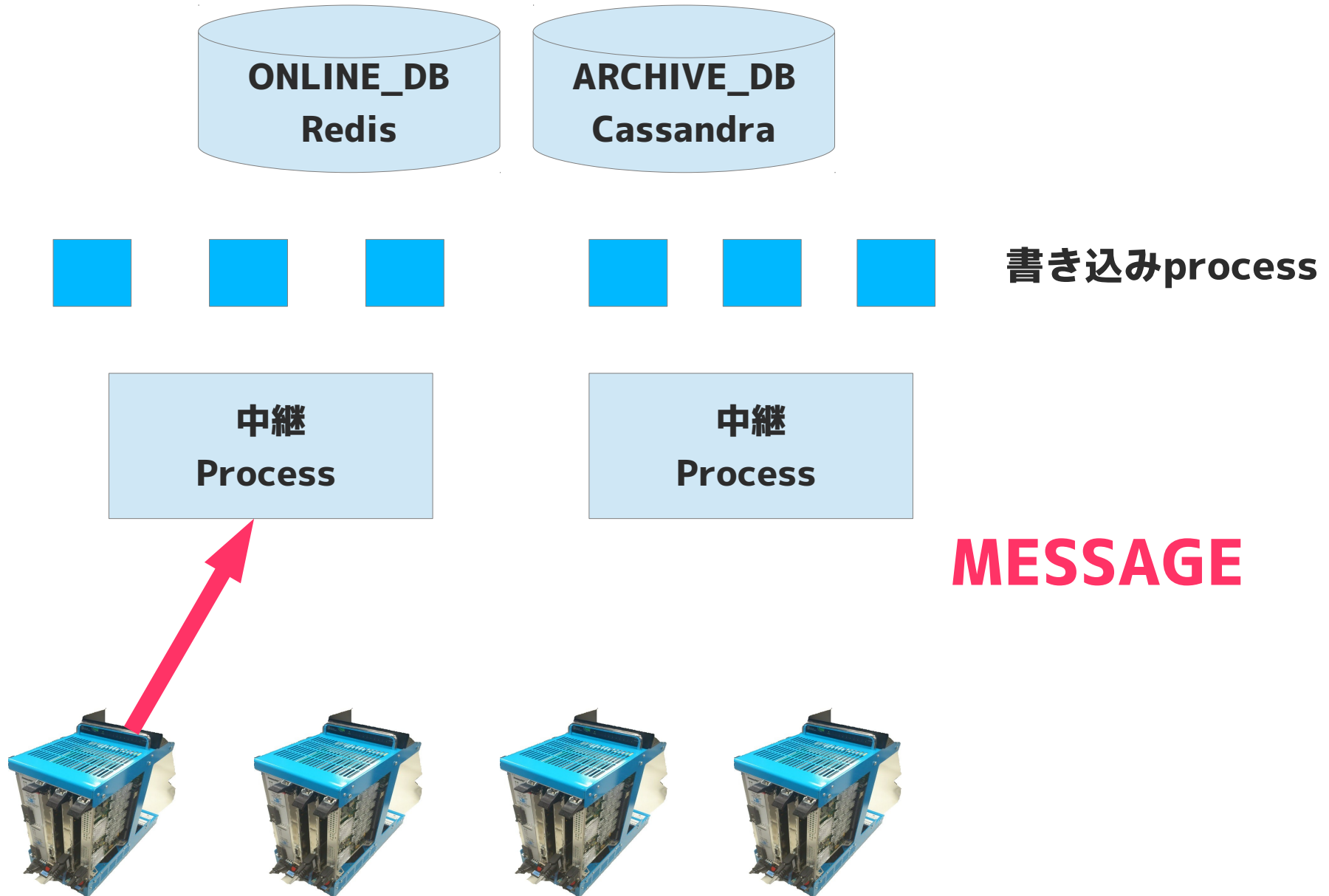
# MADOCALIIログデータ収集系 実装



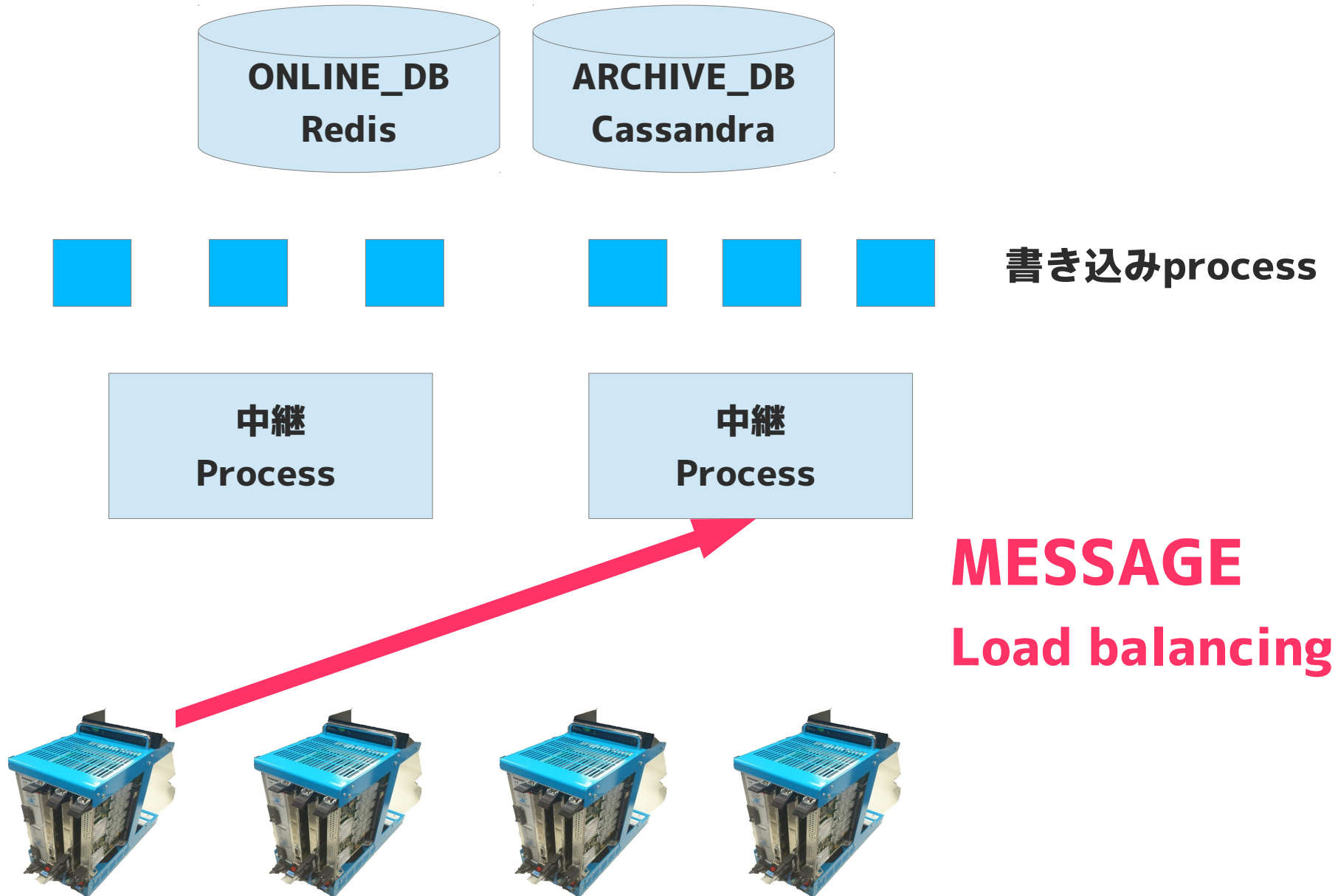
どの順番で起動  
していてもよい



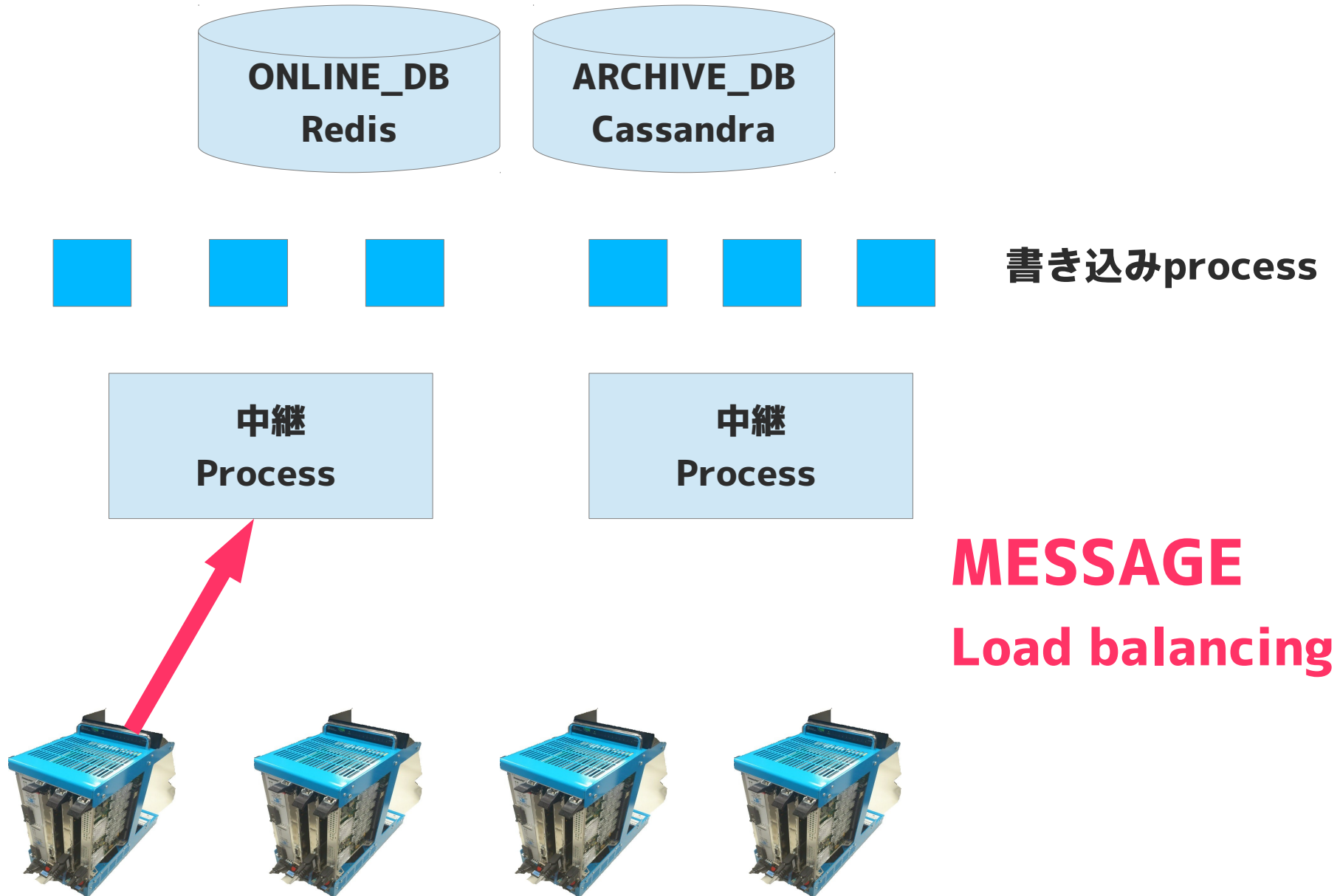
# MADOCALLログデータ収集系 実装



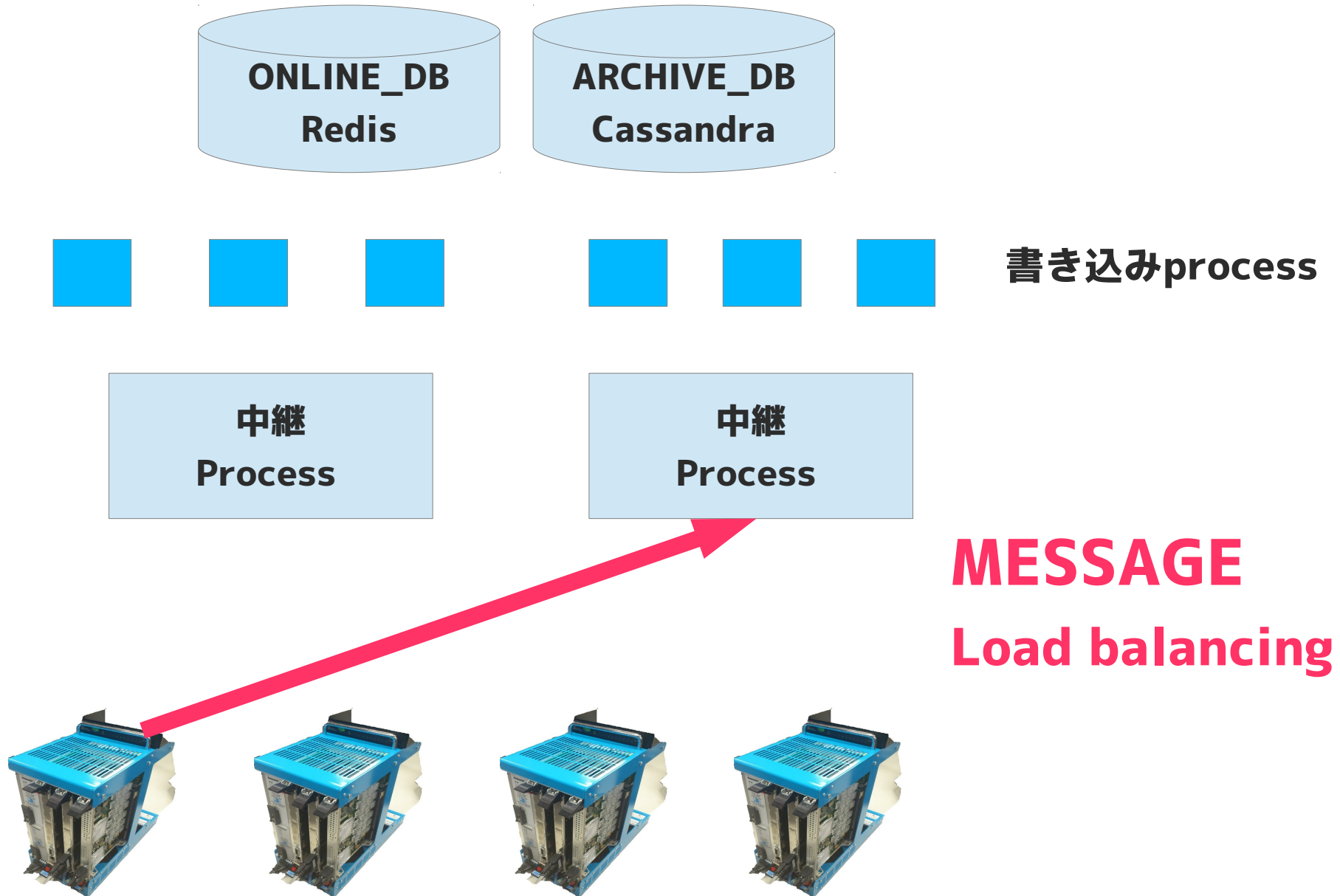
# MADOCALLログデータ収集系 実装



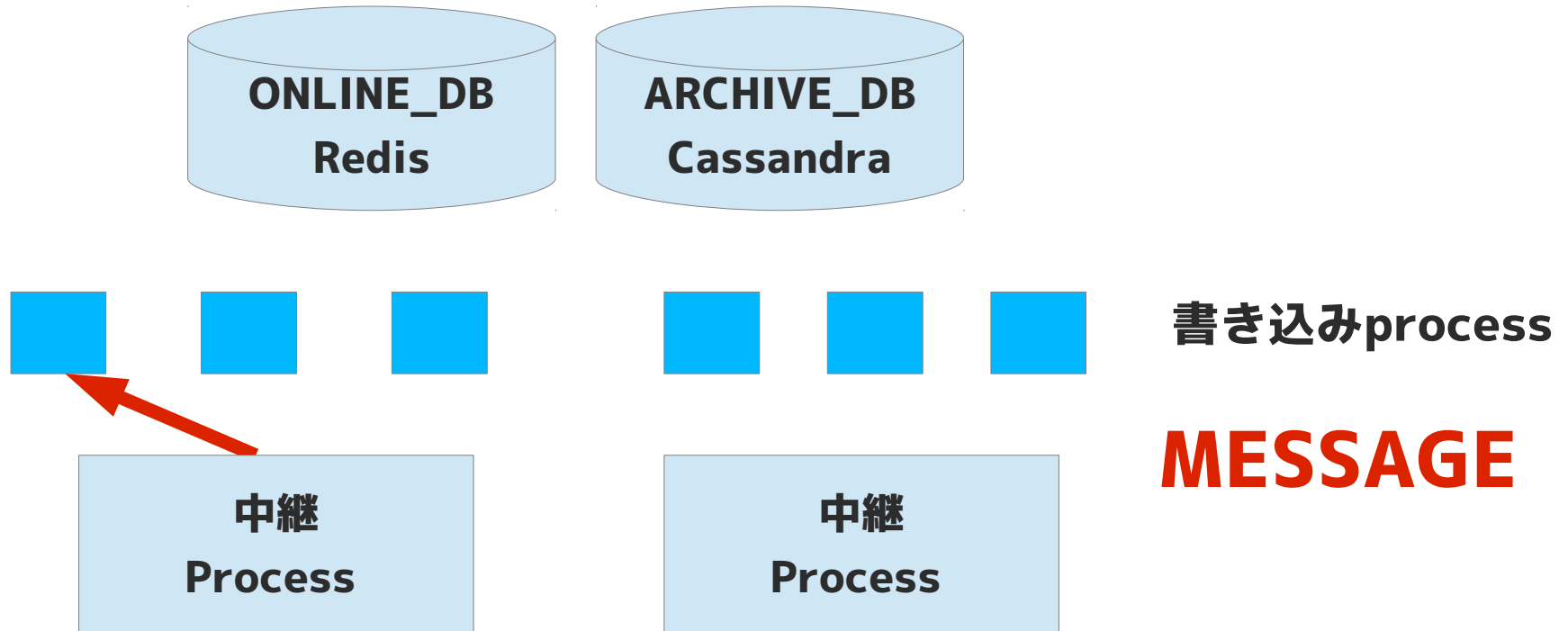
# MADOCALLログデータ収集系 実装



# MADOCALLログデータ収集系 実装

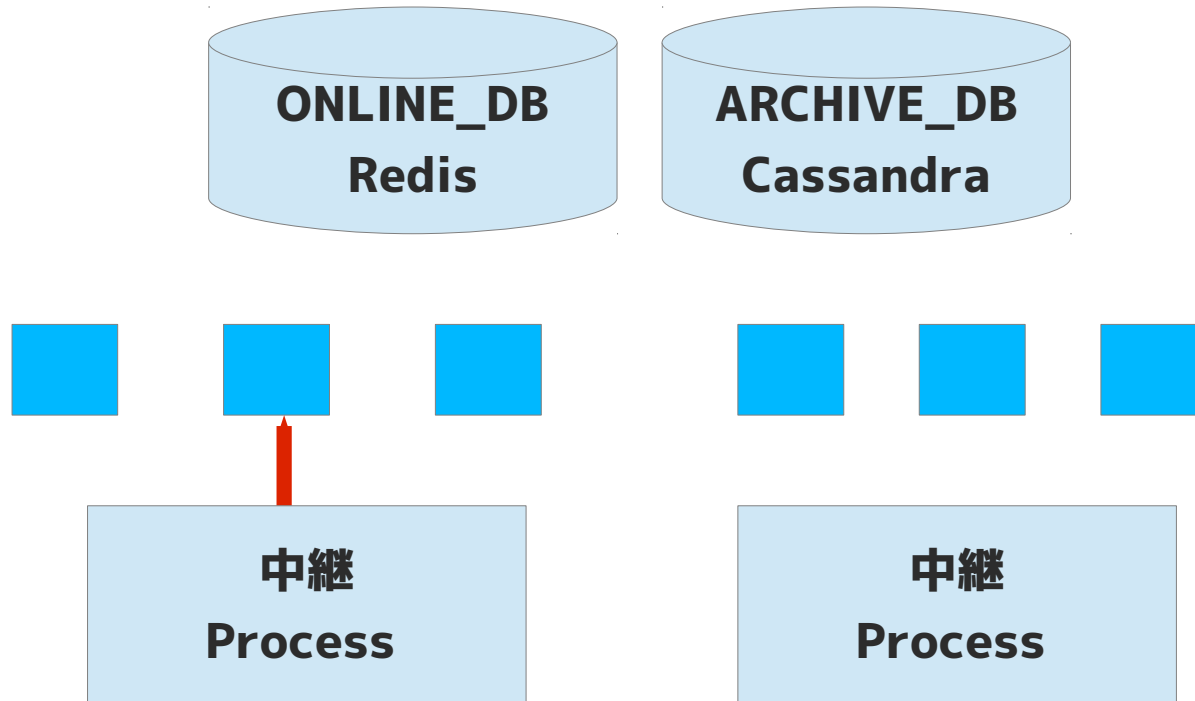


# MADOCALLログデータ収集系 実装





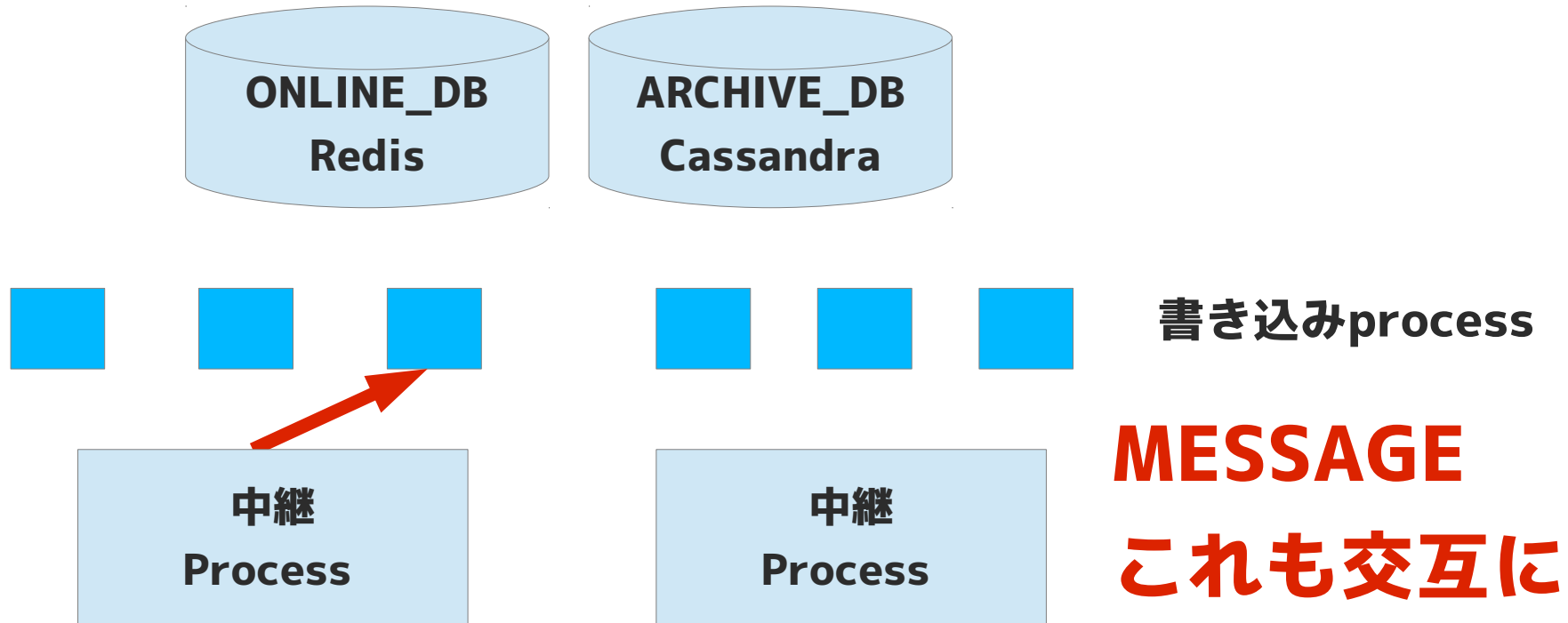
# MADOCALLログデータ収集系 実装



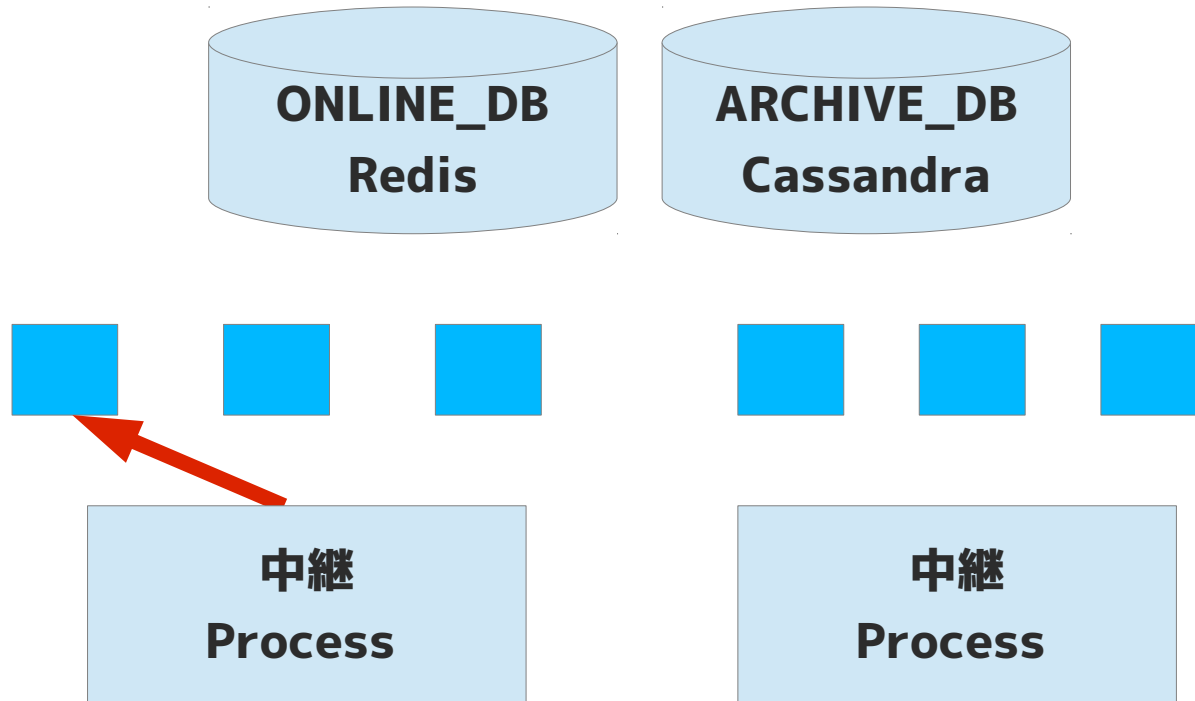
**MESSAGE**  
**これも交互に**



# MADOCALLログデータ収集系 実装



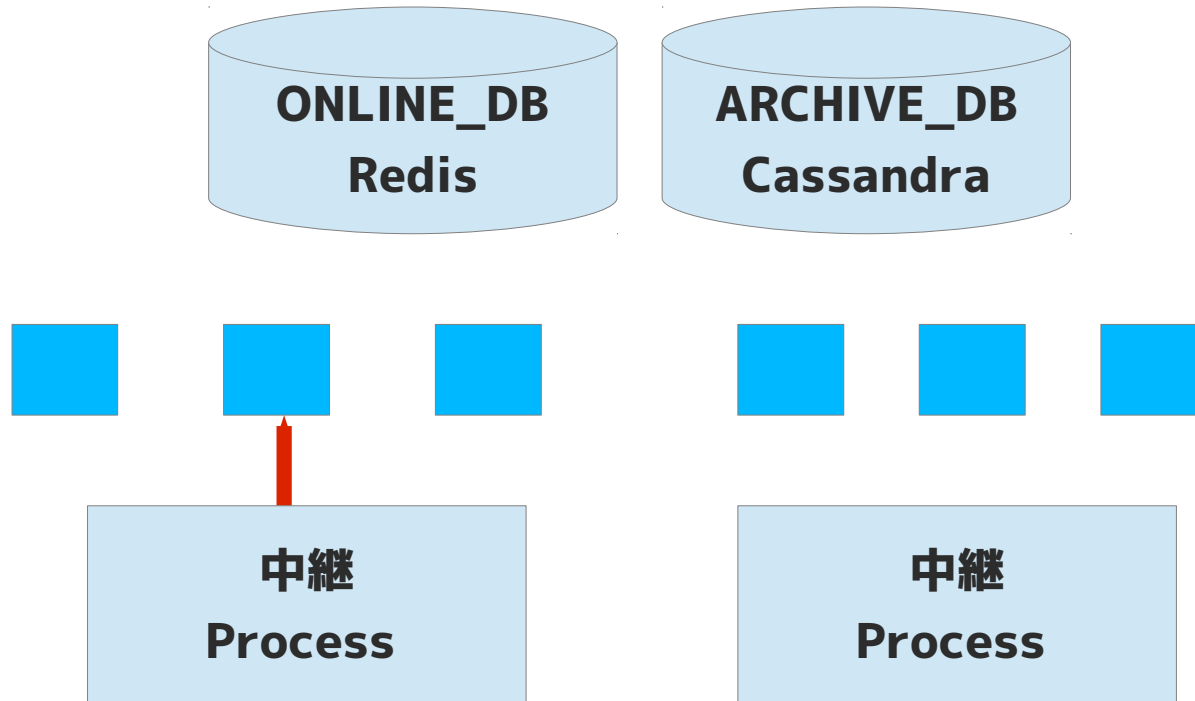
# MADOCALLログデータ収集系 実装



**MESSAGE**  
**これも交互に**



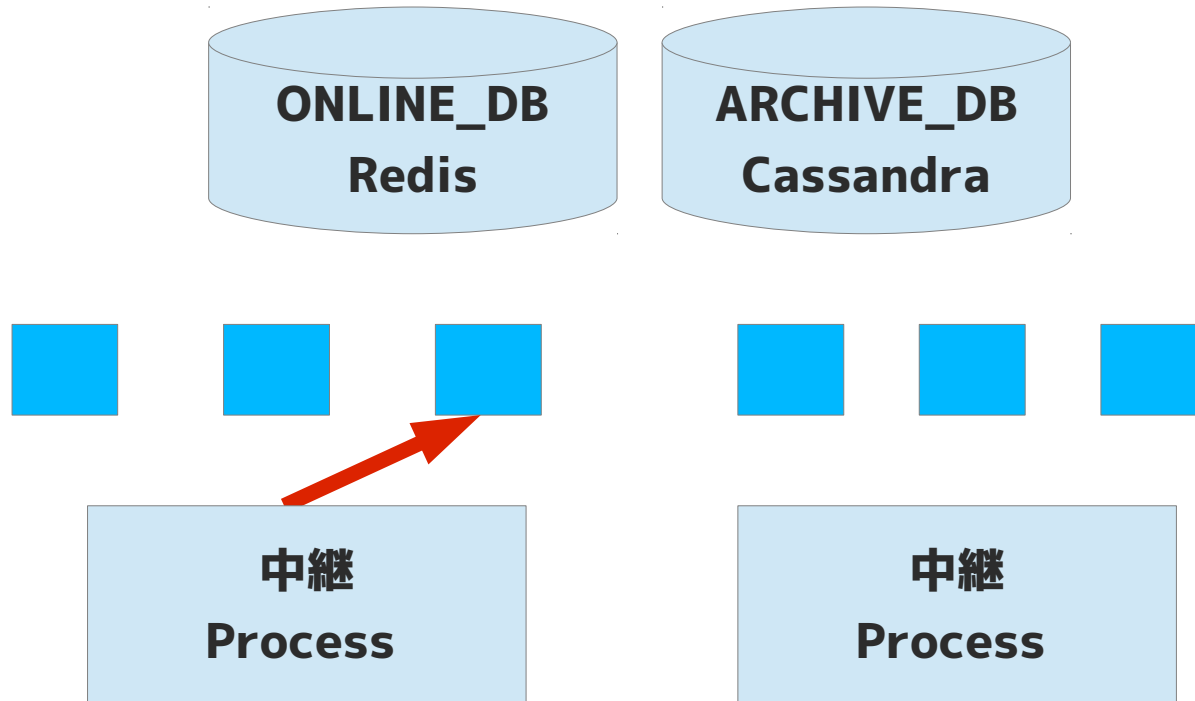
# MADOCALLログデータ収集系 実装



**MESSAGE**  
**これも交互に**



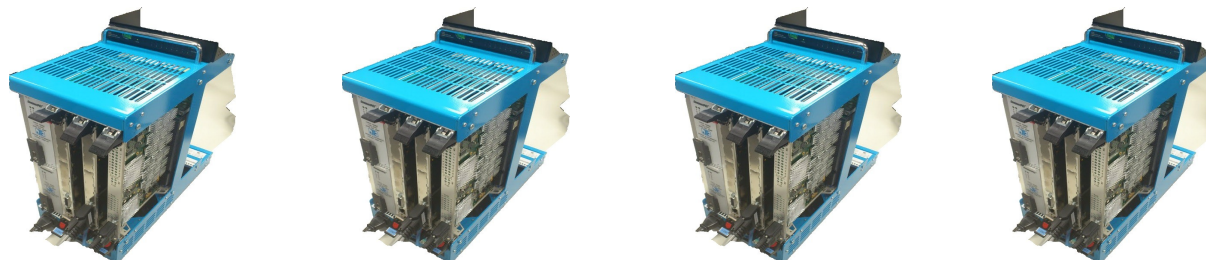
# MADOCALLログデータ収集系 実装



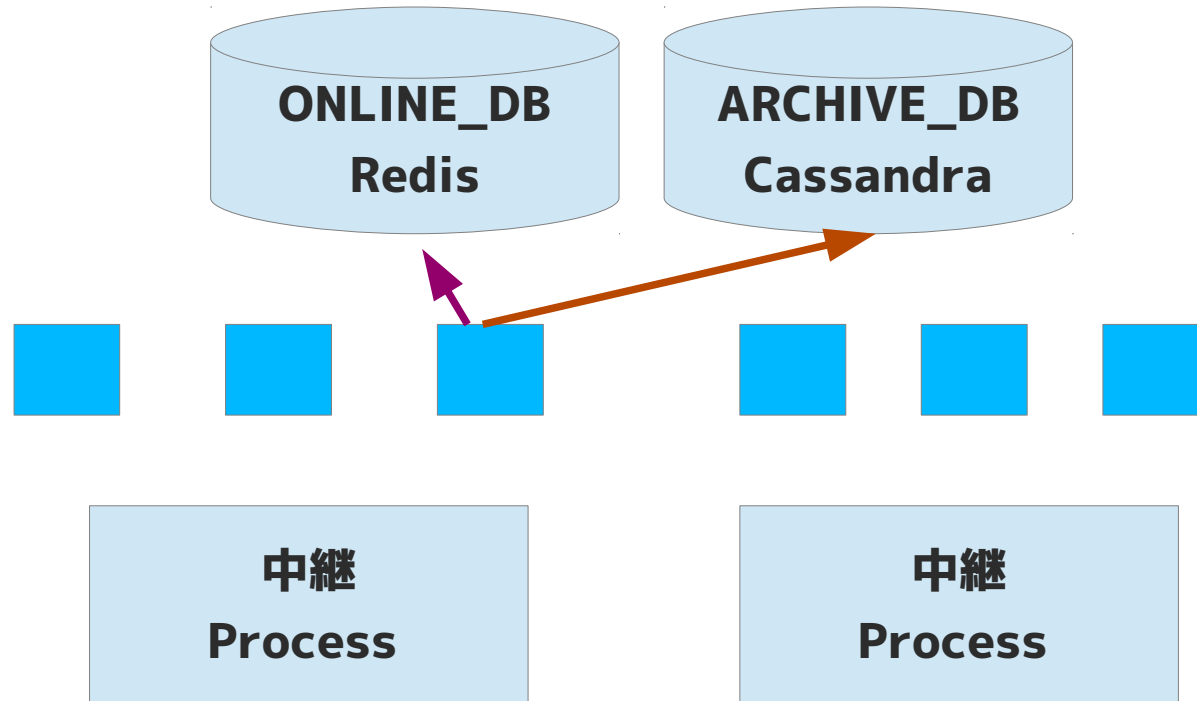
書き込みprocess

**MESSAGE**

**これも交互に**



# MADOCALLログデータ収集系 実装



Redis API

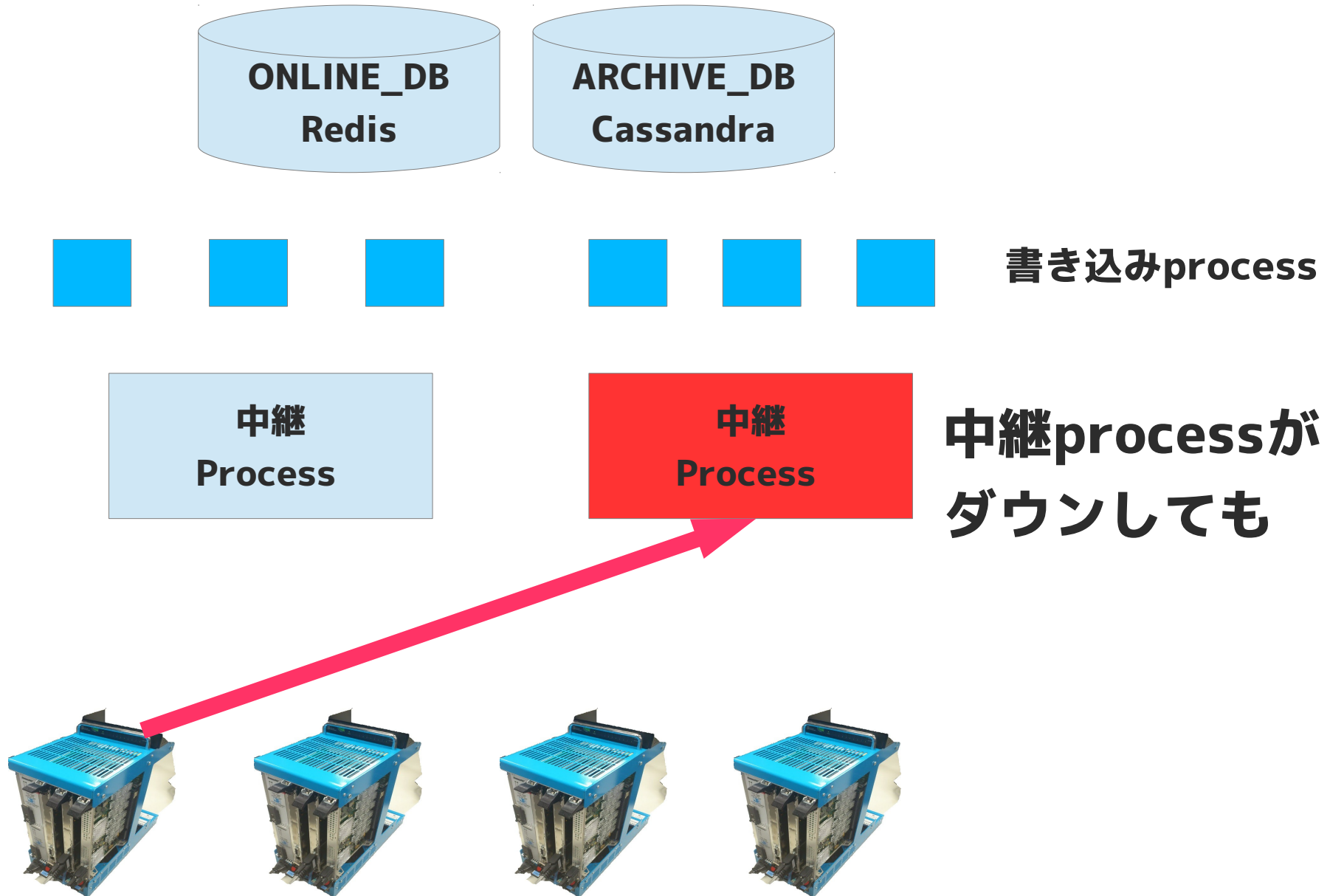
Thrift RPC

これらは同時に

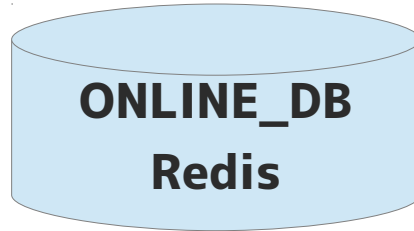
書き込みprocess



# MADOCALLログデータ収集系 実装



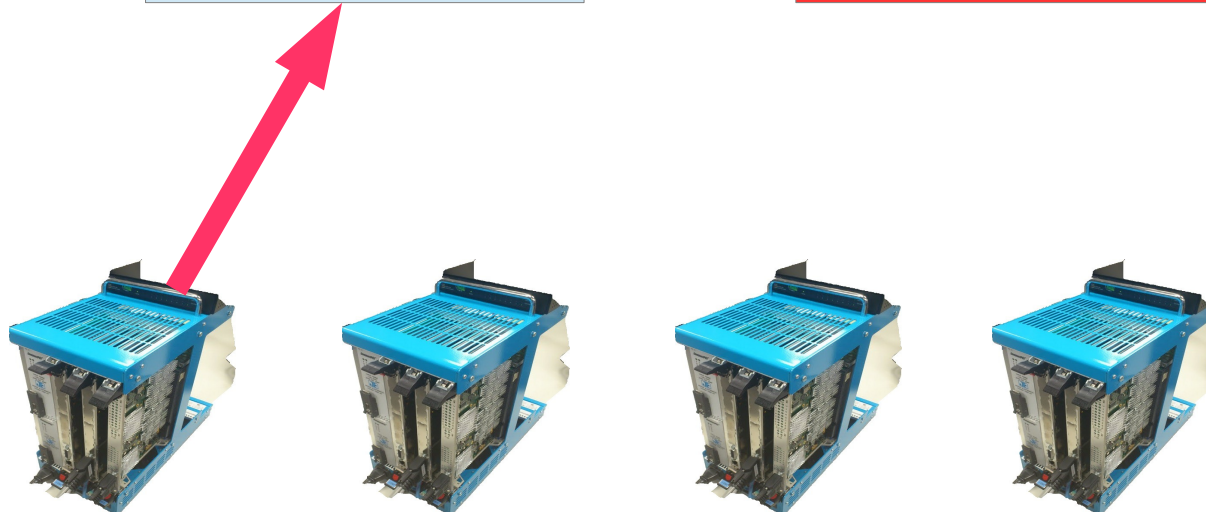
# MADOCALLログデータ収集系 実装



書き込みprocess

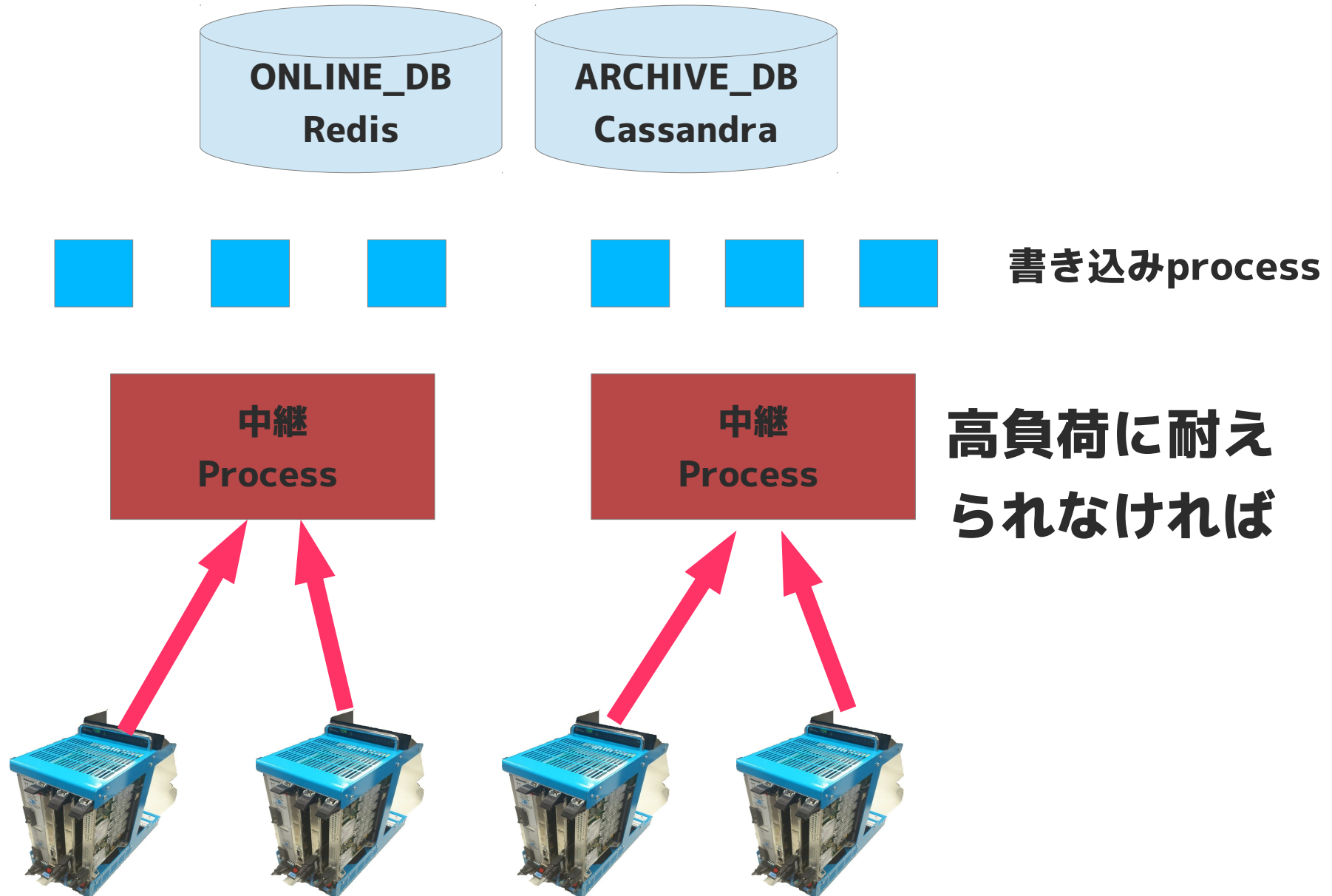


検知して  
それを避ける

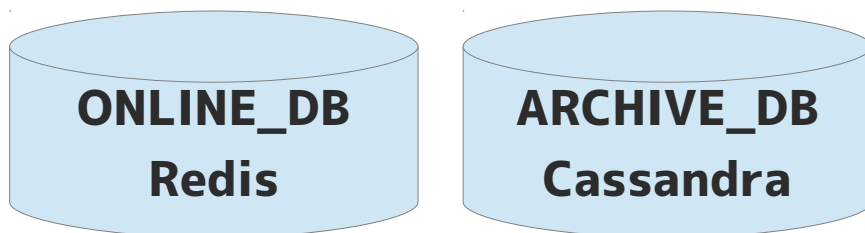




# MADOCALLログデータ収集系 実装



# MADOCAllログデータ収集系 実装



# 各パート

# 各パート

- **組込み**

# 各パート

- **組込み**

- **各データのメッセージを発行し中継サーバーに送る**

# 各パート

- **組込み**
- **中継サーバー**

# 各パート

- **組込み**
- **中継サーバー**
  - **書き込みプロセスにメッセージを中継する**

# 各パート

- **組込み**
- **中継サーバー**
  - **書き込みプロセスにメッセージを中継する**
  - **Pub/sub機構も設ける**



# 各パート

- **組込み**
- **中継サーバー**
- **書き込みプロセス**

# 各パート

- **組込み**
- **中継サーバー**
- **書き込みプロセス**
  - **メッセージを加工してデータベースに送る**

# 各パート

- **組込み**
- **中継サーバー**
- **書き込みプロセス**
  - **メッセージを加工してデータベースに送る**
  - **データベースは同期書き込みなのでプロセス数を増やして待ち行列を防ぐ**

# メッセージング

- **メッセージのロードバランシング**

# メッセージング

- **メッセージのロードバランシング**
- **接続の自由度**

# メッセージング

- **メッセージのロードバランシング**
- **接続の自由度**
- **相手の生死の判断など**

# メッセージング

- **メッセージのロードバランシング**
- **接続の自由度**
- **相手の生死の判断など**
- **こんな便利な機能は...**

# メッセージング

- **メッセージのロードバランシング**
- **接続の自由度**
- **相手の生死の判断など**
- **こんな便利な機能は**

**ZeroMQのおかげなんです**



# メッセージング

- ZeroMQ

The logo for ZeroMQ, consisting of the characters 'ØMQ' in a bold, red, sans-serif font. The 'Ø' is a circle with a diagonal slash through it, and the 'Q' has a small tail at the bottom right.

# メッセージング

- **ZeroMQ**

- **さまざまなパターンを選べる非同期のメッセージングライブラリー**

# メッセージング

- **ZeroMQ**

- **さまざまなパターンを選べる非同期のメッセージングライブラリー**
- **Many OS, Many Languages**

# メッセージング

• ZeroMQを使用  
Ada Bash Basic C Chicken Common C#  
C++ D delphi Erlang F# Felix Flex Go  
Guile Haskell Haxe Java JavaScript Julia  
LabVIEW Lua Nimrod Node.js Objective-C  
Objective ooc Perl PHP Python Q Racket  
R REBOL Red Ruby Scala Smalltalk  
Tcl XPCOM

# メッセージング

- **3part message**

Key	LGsr_mag_ps_b/current_adc:
-----	----------------------------

接頭辞(ログデーターの場合はLG)+SVOCのOC+:

# メッセージング

- **3part message**

Key	LGsr_mag_ps_b/current_adc:
-----	----------------------------

接頭辞(ログデーターの場合はLG)+SVOCのOC+:

# メッセージング

- **3part message**

Key	LGsr_mag_ps_b/current_adc:
metadata	Time:2013/04/15 13:30:11.654ns :

メタデータ：この場合は時刻のみだがマップ形式で自由に項目を追加できる。

時刻形式はunix時間\*1e9+ns

# メッセージング

- **3part message**

Key	LGsr_mag_ps_b/current_adc:
metadata	Time:2013/04/15 13:30:11.654ns :
data	3.1267123



# MADDOCA IIログデータ収集系

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**

- **信号の情報(名前、型など)は送り側だけが知っていれば良い**

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**
- **データは個別の時刻に収集できる**

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**
- **データは個別の時刻に収集できる**
- **中継サーバーにいつ接続しても、切断してもよい。**

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**
- **データは個別の時刻に収集できる**
- **中継サーバーにいつ接続しても、切断してもよい。**
- **メッセージの型は単純なので、OS、言語、データ型を選ばない。**

# MADDOCA IIログデータ収集系

- **信号管理の柔軟性**
- **データは個別の時刻に収集できる**
- **中継サーバーにいつ接続しても、切断してもよい。**
- **メッセージの型は単純なので、OS、言語、データ型を選ばない。**
- **全てのパートが簡単に追加可能**

# MADDOCA IIログデータ収集系

- 信号管理の柔軟性
- データは個別の時刻に収集できる
- 中継サーバーにいつ接続しても、切断してもよい。
- メッセージの型は単純なので、OS、言語、データ型を選ばない。
- 全てのパートが簡単に追加可能:**Scale Out**する



# MADDOCA IIデータベース実装

# MADDOCA IIデータベース実装

- 信号は個別に信号名を主キーとして蓄積

# MADDOCA IIデータベース実装

- 信号は個別に信号名を主キーとして蓄積
  - **主キー：文字列**

# MADDOCA IIデータベース実装

- 信号は個別に信号名を主キーとして蓄積
  - 主キー：文字列
  - MADDOCAの信号名は主キーに最適  
`sr_mag_ps_b/current_adc`

# MADDOCA IIデータベース実装

- 信号は個別に信号名を主キーとして蓄積
- 最新値のみ保存するONLINE\_DBと  
永久保存するARCHIVE\_DBに分離する

# MADDOCA IIデータベース実装

- 信号は個別に信号名を主キーとして蓄積
- 最新値のみ保存するONLINE\_DBと  
永久保存するARCHIVE\_DBに分離する
- NoSQL (Not Only SQL)技術を取り入れRDB  
では得られなかった機能、性能を得る。

# ONLINE\_DB

# ONLINE\_DB

- Redis





# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**

# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**
- **Key value型**

# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**
- **Key value型**
  - **Key : LGsr\_mag\_ps\_b/current**
  - **Value: メタデータ+valueを文字列化**

# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**
- **Key value型**
- **最新値+時刻のみの保存**

# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**
- **Key value型**
- **最新値+時刻のみの保存**
- **Keyのhash値でサーバーを分散化した**

# ONLINE\_DB

- **Redis**

- **高速インメモリーデータベース**
- **Key value型**
- **最新値+時刻のみの保存**
- **Keyのhash値でサーバーを分散化した**
- **多数のkey-value対を一挙取得も可**

# ARCHIVE\_DB

# ARCHIVE\_DB

- Apache Cassandra



A highly scalable, eventually consistent, distributed, structured key-value store.



# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

sig1:20130504

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0
	value0

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1
	value0	value1

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2
	value0	value1	value2

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2	t3
	value0	value1	value2	value3

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2	t3	t4
	value0	value1	value2	value3	value4

# ARCHIVE\_DB

- Apache Cassandra

- カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2	t3	t4
	value0	value1	value2	value3	value4

**1つの行キーで1日分のデータを保存する**



# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2	t3	t4
	value0	value1	value2	value3	value4
sig2:20130504	t0	t1			
	value0	value1			

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース

行キー

カラムキー

sig1:20130504	t0	t1	t2	t3	t4
	value0	value1	value2	value3	value4
sig2:20130504	t0	t1			
	value0	value1			
sig3:20130504	t0	t1	t2	t3	
	value0	value1	value2	value3	

# ARCHIVE\_DB

- **Apache Cassandra**
  - **カラム指向データベース**
  - **分散型**

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース
  - 分散型
    - データ冗長化

# ARCHIVE\_DB

- Apache Cassandra
    - カラム指向データベース
    - 分散型
      - データ冗長化
- 今回は3重

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース
  - 分散型
    - データ冗長化
    - 今回は3重
      - 高信頼性

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース
  - 分散型
    - データ冗長化
    - Scale outする

# ARCHIVE\_DB

- **Apache Cassandra**
  - **カラム指向データベース**
  - **分散型**
  - **マスターノード無し**



# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース
  - 分散型
  - マスターノード無し
    - 単一障害点が無い

# ARCHIVE\_DB

- Apache Cassandra
  - カラム指向データベース
  - 分散型
  - マスターノード無し
    - 単一障害点が無い
    - 高信頼性

# ARCHIVE\_DB

- **Apache Cassandra**
  - **カラム指向データベース**
  - **分散型**
  - **マスターノード無し**
  - **データー値はMessagePackによる文字列**

# 書き込みテスト

- Intel Xeon X3470
- 2.93GHz 4Core 16
- CentOS 6.2 64bit
- Redis 2.6.10
- Cassandra 1.15
  - OracleJava JVM 1.6.0
- 信号数 **47397**
- **1Hz** : 現行の **6倍**
- 平均キー長 **38.7 byte**
- メタデータ **13bytes**
- データー書き込み**240プロセス**
- 中継サーバー**3**
- 書き込みプロセス/server **8**
- Redis process **4**
- Cassandra server **6**

# 書き込み長期テスト

# 書き込み長期テスト

- 3ヶ月連続運転

# 書き込み長期テスト

- 3ヶ月連続運転
- データー抜け無し

# 書き込み長期テスト

- **3ヶ月連続運転**
- **データー抜け無し**
- **ノードを落すテスト**



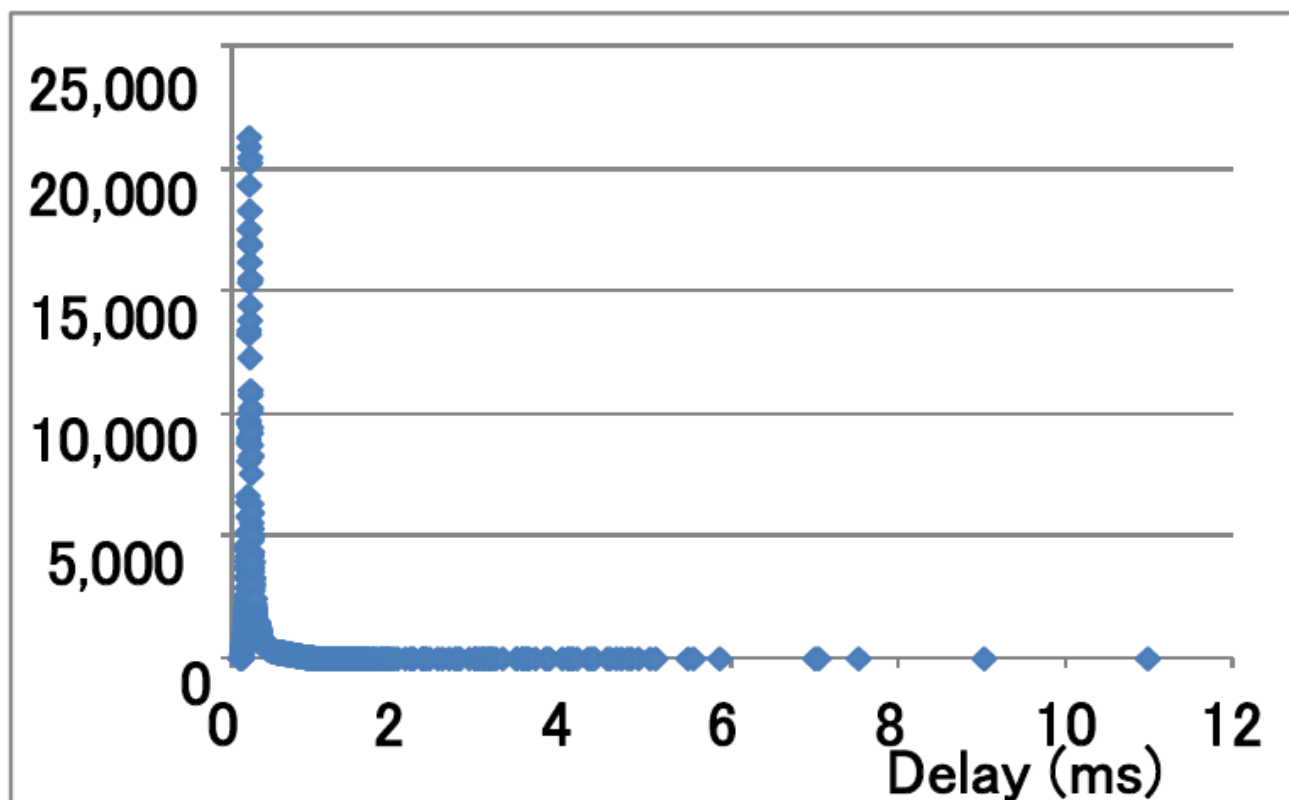
# 書き込み長期テスト

- **3ヶ月連続運転**
- **データー抜け無し**
- **ノードを落すテスト**
  - **データー抜けやりカバリーに問題なし**

# 読み込みテスト

- **読み込みテストは書き込みテストと並行しておこなった。**
  - **実環境に合わせるため**

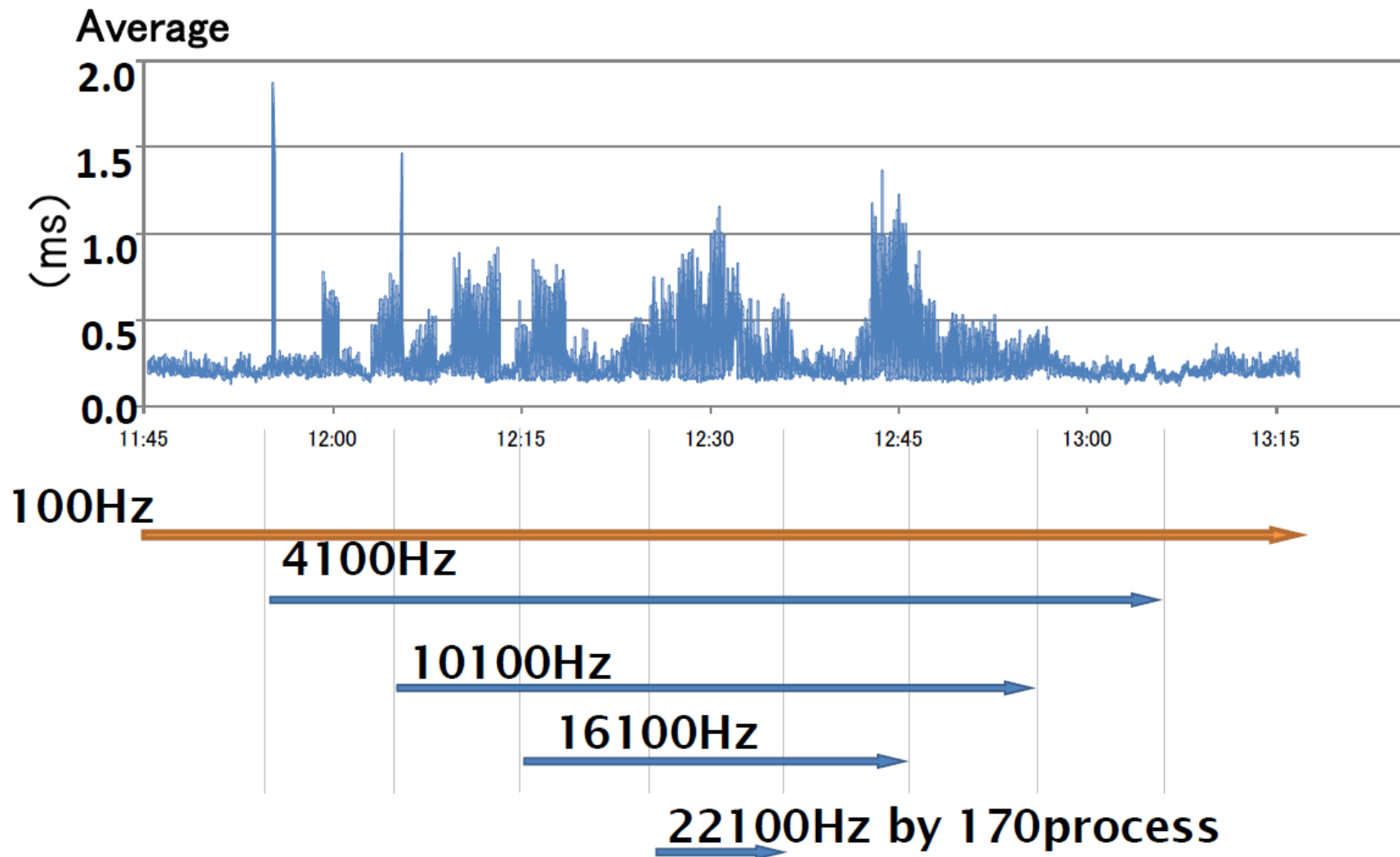
# ONLINE\_DB 読み込み



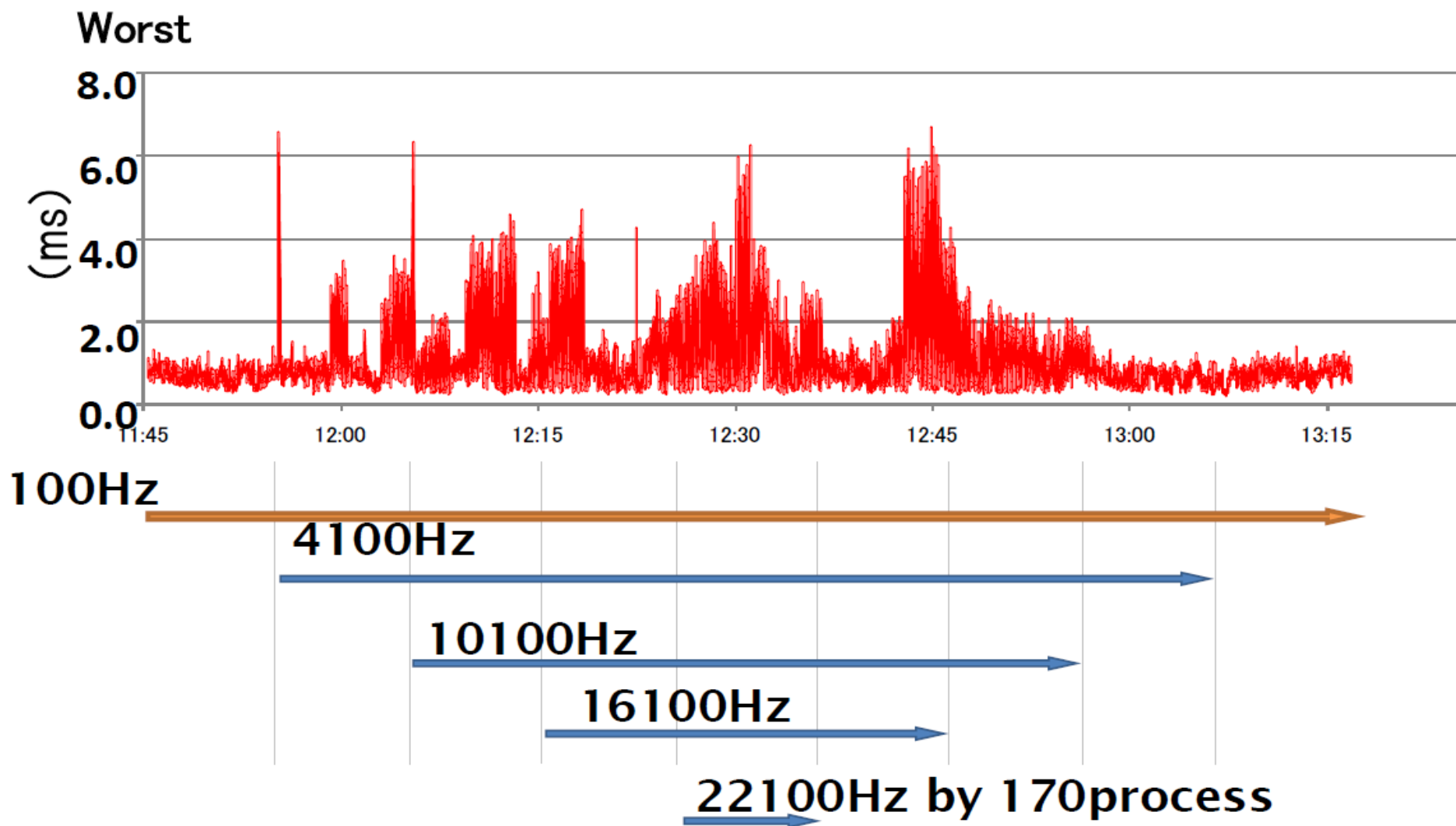
	ms
最小値	0.09
最大値	10.98
平均値	0.26
標準偏差	0.14

# ONLINE\_DB 読み込み

- 時間経過とともにテストプロセスを増減



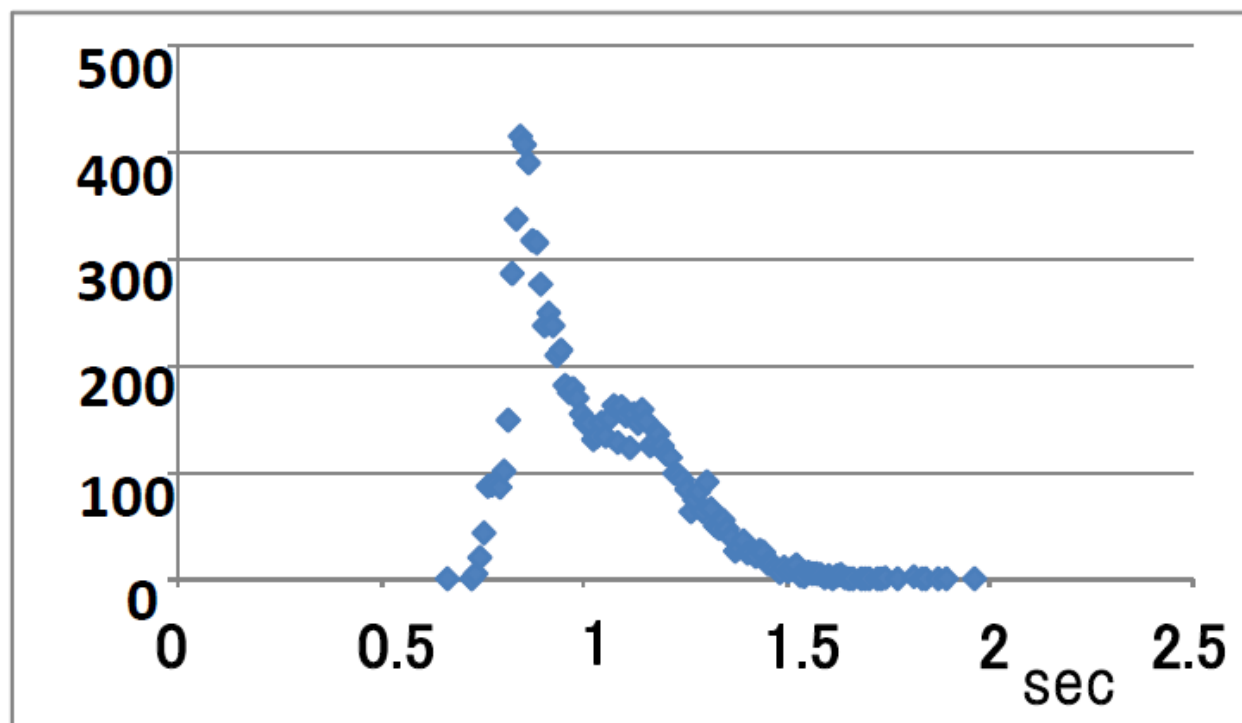
# ONLINE\_DB 読み込み 最悪値



# ARCHIVE\_DB 読み込み

- 1秒ごとの1日分のデータを10,000回計測

86,400 点



	単位 : sec
最小値	0.66
最大値	1.96
平均値	1.01
標準偏差	0.18

# これからの計画

# これからの計画

- **2013年度一部本番環境でテスト**



# これからの計画

- **2013年度一部本番環境でテスト**
  - **RDBと並行して運用**

# これからの計画

- **2013年度一部本番環境でテスト**
  - **RDBと並行して運用**
- **2014年度本格投入**

# これからの計画

- **2013年度一部本番環境でテスト**
  - **RDBと並行して運用**
- **2014年度本格投入**
  - **複雑なデータ型 (BPM等、array,map) も移行**

# 結論

# 結論

- NoSQLやメッセージングをとりいれることで  
今までのMADOCAより

# 結論

- NoSQLやメッセージングをとり入れることで  
今までのMADOCAより
  - シンプルなデータ管理

# 結論

- NoSQLやメッセージングをとりいれることで  
今までのMADOCAより
  - シンプルなデータ管理
  - 柔軟性

# 結論

- NoSQLやメッセージングをとりいれることで  
今までのMADOCAより

シンプルなデータ管理

- 柔軟性
- 高信頼性



# 結論

- **NoSQLやメッセージングをとりいれることで  
今までのMADOCAより**
  - **シンプルなデータ管理**
  - **柔軟性**
  - **高信頼性**
  - **高性能**

# 結論

- NoSQLやメッセージングをとりいれることで  
今までのMADOCAより

## シンプルなデータ管理

- 柔軟性
- 高信頼性
- 高性能
- 高拡張性

# 結論

- **NoSQLやメッセージングをとりいれることで  
今までのMADOCAより**
  - **シンプルなデータ管理**
  - **柔軟性**
  - **高信頼性**
  - **高性能**
  - **高拡張性**
  - **低コスト を実現できた。**

# 結論

- **NoSQLやメッセージングをとりいれることで  
今までのMADOCAより**
  - シンプルなデータ管理
  - 柔軟性
  - 高信頼性
  - 高性能
  - 高拡張性
  - 低コスト
- **テストをさらにすすめ、本格導入を目指す**