FUJITSU
shaping tomorrow with you

SC研究会
決定木を用いたクラウドインフラ設定パラメータの自動生成

2013月6月17日　16:50 - 17:15

株式会社富士通研究所　システムソフトウェア研究所
システムマネジメント研究部
○内海哲哉，北島信哉，菊池慎司，松本安英

目次

- 背景
- 課題
- 既存手法
  - クラスタリングを用いるパラメータ設計パターン抽出
- 提案手法
  - パラメータ推定
  - 再帰的クラスタリングを用いる設計手順生成
  - パラメータ自動設計
- 評価
- 結果
- 考察
- まとめ

背景

FUJITSU

- **クラウドコンピューティング技術**
  - コスト削減やリソース使用効率改善に貢献
- **多くの企業で採用＆ビジネスに利用**
  - Amazon web services, Google Cloud Platform, Windows Azure, NTT DATA, HITACHI, NEC, FUJITSU, etc...

- **クラウド市場は拡大を続ける**
  - 需要の増大
  - クラウドベンダーは，インフラを素早く正確に構築する必要がある

クラウドインフラの構築と構成

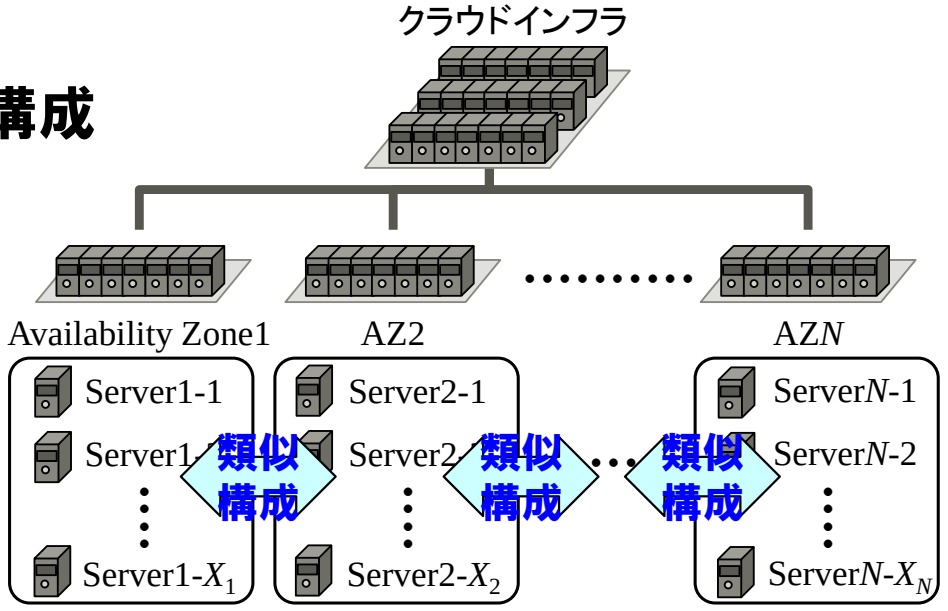
FUJITSU

■構築の特徴

- ■コスト削減のため同様の構造が繰り返し用いられる

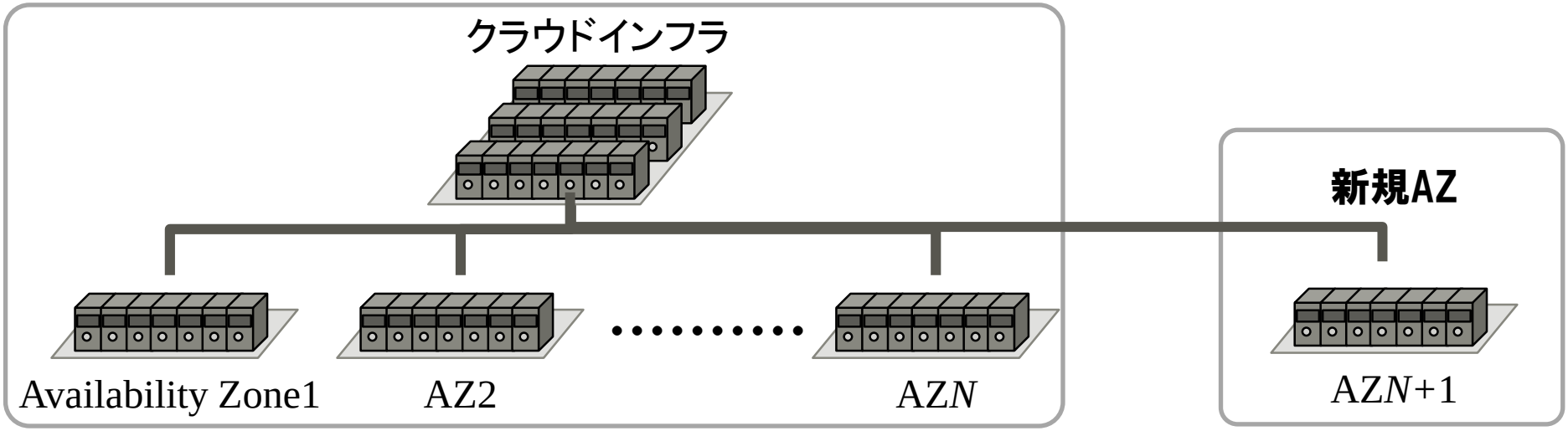
■構成の特徴

- ■複数のデータセンタによって構成
- ■データセンタ：それぞれ特定の場所AZ（Availability Zone）に構築
- ■Availability Zone
  - 数十～数百のサーバによって構成
  - 各々類似の構成をしている



課題

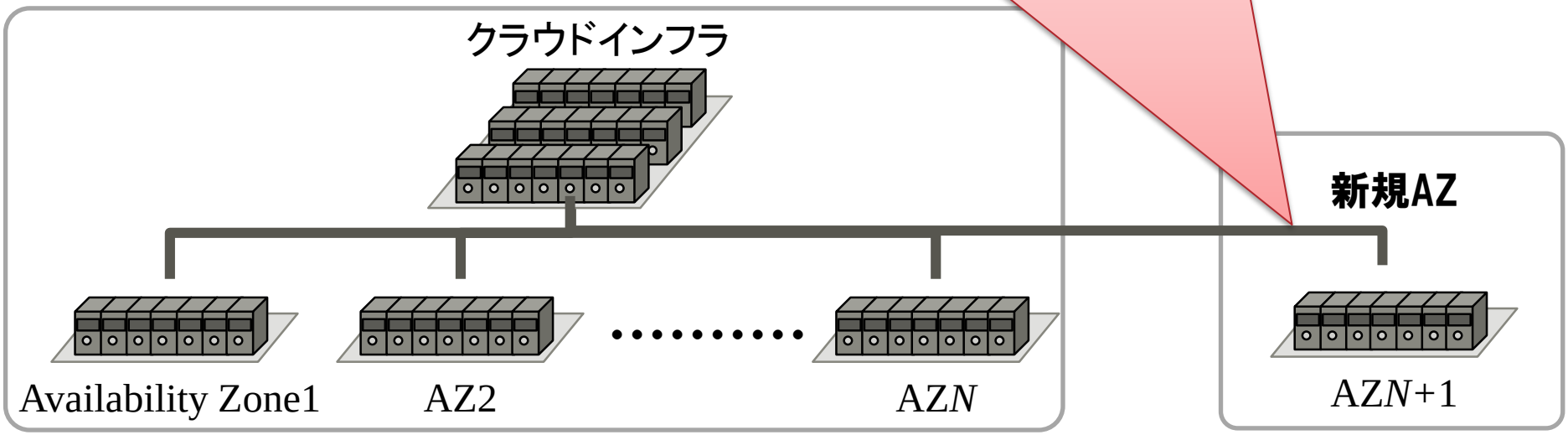
- 大規模クラウドインフラ
  - 多数のサーバやストレージ，ミドルウェアなどから構成
- 膨大な数のパラメータが存在
  - 数万～数十万にも及ぶ
  - パラメータ：IPアドレス等のOS設定値やミドルウェアのパラメータ



課題

- 大規模クラウドインフラ
  - 多数のサーバやストレージ，ミドルウェアなどから構成
- 膨大な数のパラメータが存在
  - 数万
  - パラ
    値やミドルウェアのパラメータ

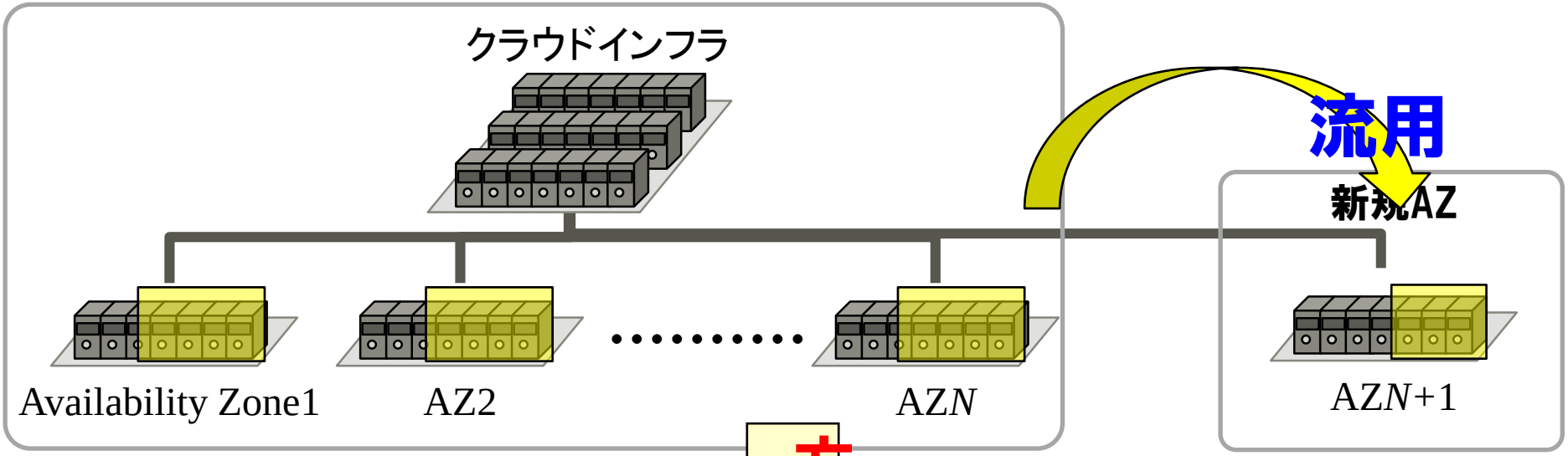
人手で正しく設計することは非常に困難



いままで

FUJITSU

- パラメータを自動で設計する技術が提案されている
  - ①複数の既存システムに共通する設定のパターンを抽出
  - ②抽出したパターンを新規に構築するインフラの設定値に反映



一方

”共通部分に該当しないパラメータ”

⇒マニュアル入力が必要

例）共通と非共通

FUJITSU

■複数のシステム（AZ）で共通

同じ

AZ0	server01	server02	server03	server04	server05
UTC	FALSE	FALSE	TRUE	TRUE	TRUE
LANG	Japanese	Japanese	English	English	English

AZ1	server01	server02	server03	server04	server05
UTC	FALSE	FALSE	TRUE	TRUE	TRUE
LANG	Japanese	Japanese	English	English	English

同じ

■複数のシステム（AZ）で非共通

異なる

AZ0	server01	server02	server03	server04	server05
IPアドレス	172.16.0.1	172.16.0.2	172.16.0.3	172.16.0.4	172.16.0.5
Gateway	10.0.0.1				10.0.0.1

バラバラな数値

AZ1	server01	server02	server03	server04	server05
IPアドレス	172.16.1.1	172.16.1.2	172.16.1.3	172.16.1.4	172.16.1.5
Gateway	10.0.1.1	10.0.1.1	10.0.1.1	10.0.1.1	10.0.1.1

異なる

提案手法

FUJITSU

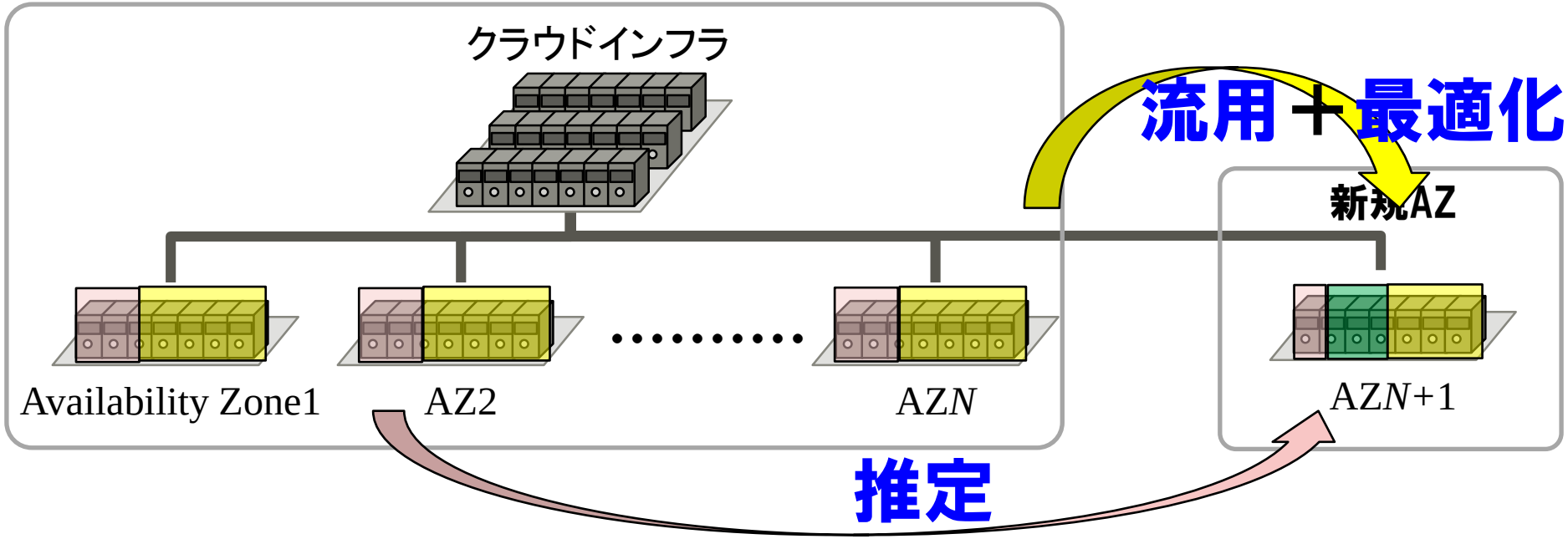
■ 共通部分に該当しないパラメータの設計パターンも特定

既存手法（12’ uchiumi, et al）

＋

パラメータ推定

再帰的クラスタリングを用いる
設計手順生成
（パターン最適化）



提案手法

FUJITSU

■共通部分に該当しないパラメータの設計パターンも特定

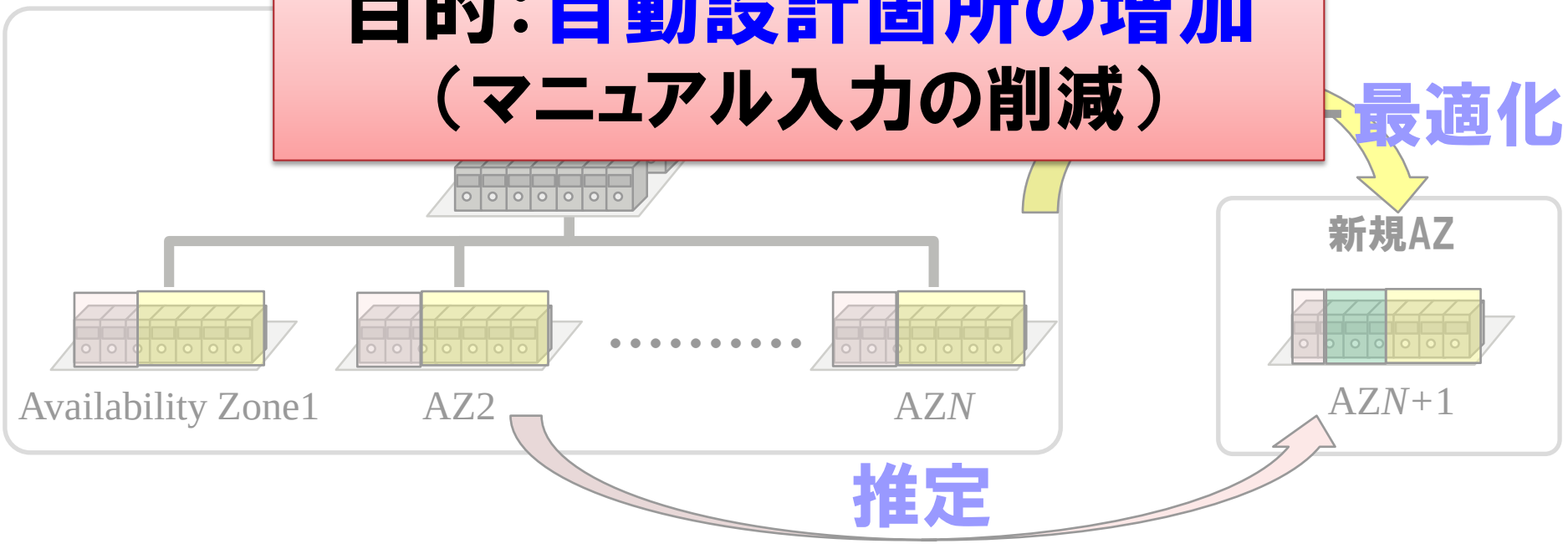
既存手法（12’内海ら）

＋

パラメータ推定

再帰的クラスタリングを用いる
設計手順生成
（パターン最適化）

目的：自動設計箇所の増加
（マニュアル入力の削減）



提案手法概要

FUJITSU

入力（教師データ）

既存システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N

AZ_2	Server2-1	Server2-2	…	Server2-S_2

AZ_1	Server1-1	Server1-2	…	Server1-S_1
Parameter1	$V_{11}(1)$	$V_{12}(1)$	…	$V_{1S_1}(1)$
Parameter2	$V_{11}(2)$	$V_{12}(2)$	…	$V_{1S_1}(2)$
Parameter3	$V_{11}(3)$	$V_{12}(3)$	…	$V_{1S_1}(3)$
…	…	…	…	…
ParameterP_{11}	$V_{11}(P_{11})$	$V_{12}(P_{12})$	…	$V_{1S_1}(P_{1S_1})$

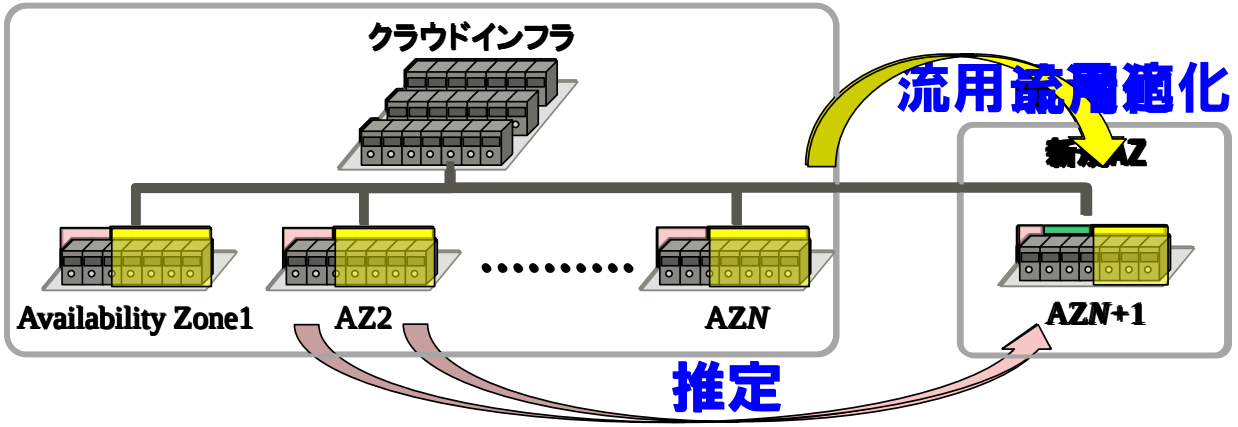
出力

新規システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N
Parameter1	$V_{N1}(1)$	$V_{N2}(1)$	…	$V_{NS_N}(1)$
Parameter2	$V_{N1}(2)$	$V_{N2}(2)$	…	$V_{NS_N}(2)$
Parameter3	$V_{N1}(3)$	$V_{N2}(3)$	…	$V_{NS_N}(3)$
…	…	…	…	…
ParameterP_{N1}	$V_{N1}(P_{N1})$	$V_{N2}(P_{N2})$	…	$V_{NS_N}(P_{NS_N})$

提案手法

既存手法 → パラメータ推定 → 再帰的クラスタリングを用いる設計手順生成



既存手法について

FUJITSU

入力（教師データ）

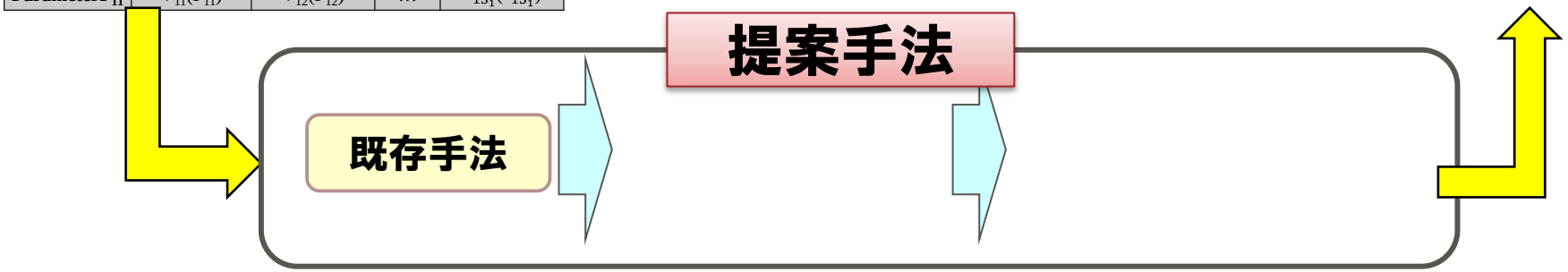
既存システム構成情報

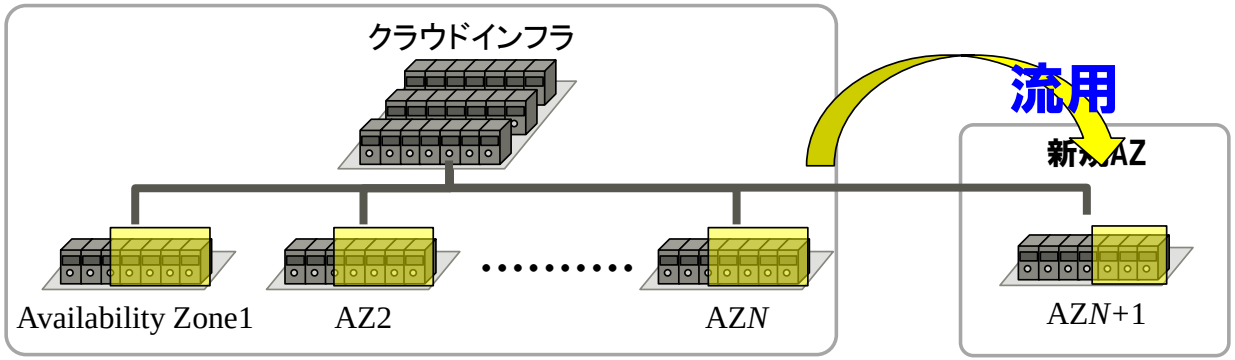
AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N
AZ_2	Server2-1	Server2-2	…	Server2-S_2
AZ_1	Server1-1	Server1-2	…	Server1-S_1
Parameter1	$V_{11}(1)$	$V_{12}(1)$	…	$V_{1S_1}(1)$
Parameter2	$V_{11}(2)$	$V_{12}(2)$	…	$V_{1S_1}(2)$
Parameter3	$V_{11}(3)$	$V_{12}(3)$	…	$V_{1S_1}(3)$
…	…	…	…	…
ParameterP_{11}	$V_{11}(P_{11})$	$V_{12}(P_{12})$	…	$V_{1S_1}(P_{1S_1})$

出力

新規システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N
Parameter1	$V_{N1}(1)$	$V_{N2}(1)$	…	$V_{NS_N}(1)$
Parameter2	$V_{N1}(2)$	$V_{N2}(2)$	…	$V_{NS_N}(2)$
Parameter3	$V_{N1}(3)$	$V_{N2}(3)$	…	$V_{NS_N}(3)$
…	…	…	…	…
ParameterP_{N1}	$V_{N1}(P_{N1})$	$V_{N2}(P_{N2})$	…	$V_{NS_N}(P_{NS_N})$





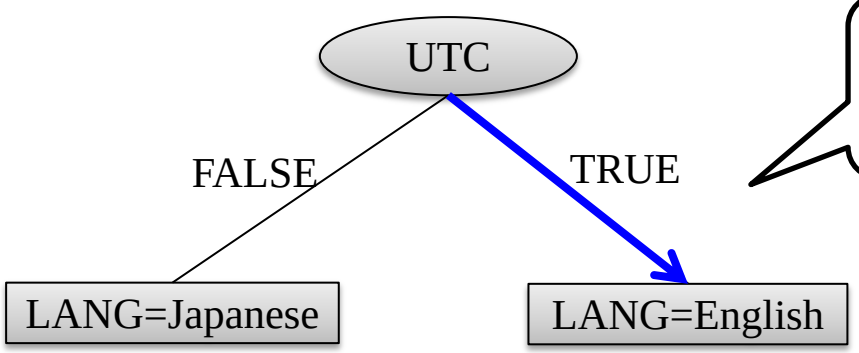
既存手法

FUJITSU

- **既存手法：以前提案したパラメータ検証技術**（12’ uchiumi, et al）
  - 稼働中のクラウドから設定の傾向を”設計パターン”として抽出

- **決定木解析による設計パターン抽出**
  - 多数の対象をエントロピー（情報量）によって分別

	server01	server02	server03	server04	server05
UTC	FALSE	FALSE	TRUE	TRUE	TRUE
LANG	Japanese	Japanese	English	English	English



大多数に共通するパターンを
IF-THENルールとして抽出

IF “UTC＝TRUE“
THEN “LANG=English”

パラメータ推定について

FUJITSU

入力（教師データ）

既存システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N

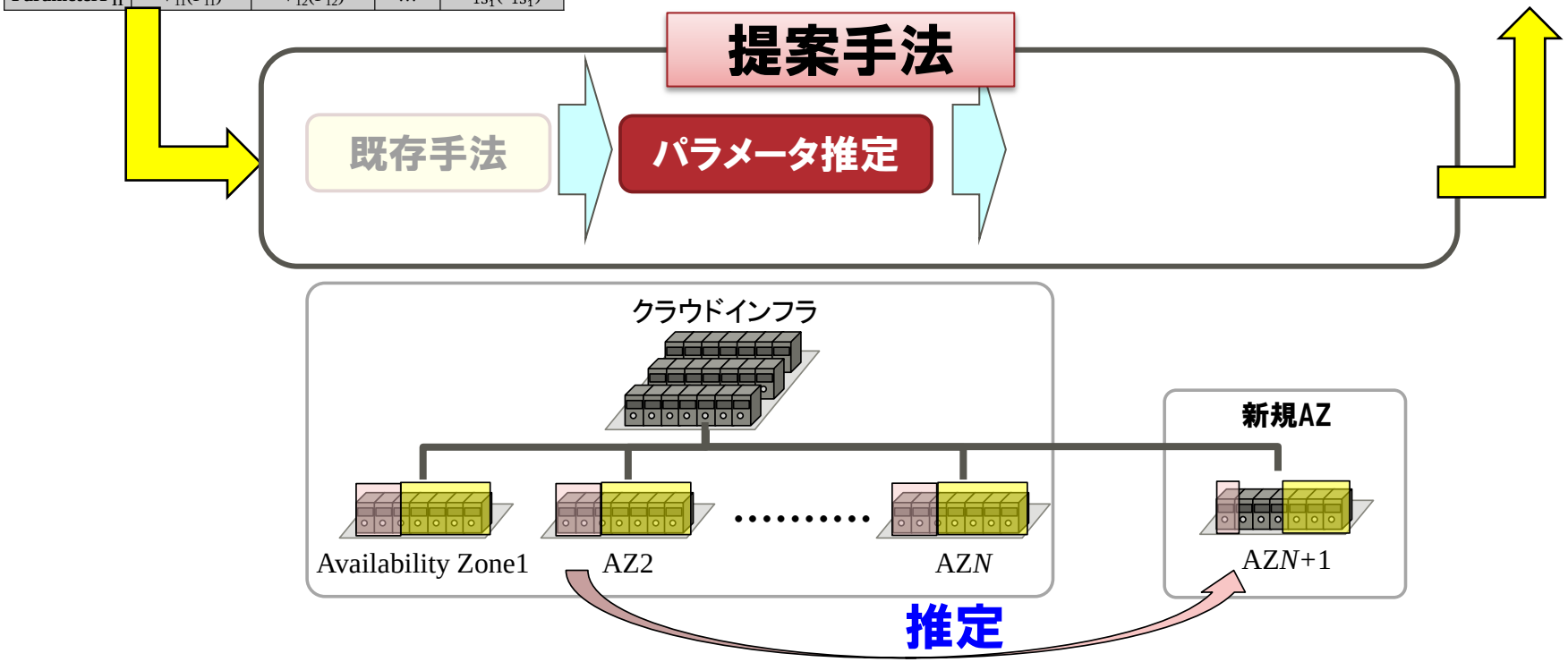
AZ_2	Server2-1	Server2-2	…	Server2-S_2

AZ_1	Server1-1	Server1-2	…	Server1-S_1
Parameter1	$V_{11}(1)$	$V_{12}(1)$	…	$V_{1S_1}(1)$
Parameter2	$V_{11}(2)$	$V_{12}(2)$	…	$V_{1S_1}(2)$
Parameter3	$V_{11}(3)$	$V_{12}(3)$	…	$V_{1S_1}(3)$
…	…	…	…	…
ParameterP_{11}	$V_{11}(P_{11})$	$V_{12}(P_{12})$	…	$V_{1S_1}(P_{1S_1})$

出力

新規システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N
Parameter1	$V_{N1}(1)$	$V_{N2}(1)$	…	$V_{NS_N}(1)$
Parameter2	$V_{N1}(2)$	$V_{N2}(2)$	…	$V_{NS_N}(2)$
Parameter3	$V_{N1}(3)$	$V_{N2}(3)$	…	$V_{NS_N}(3)$
…	…	…	…	…
ParameterP_{N1}	$V_{N1}(P_{N1})$	$V_{N2}(P_{N2})$	…	$V_{NS_N}(P_{NS_N})$



パラメータ推定

FUJITSU

- 決定木解析ではパターンを抽出できないパラメータ
  - 全サーバで固有の設定値をもつパラメータ
    - IPアドレス，ホストネームなど
  - 各データセンタ（AZ）で固有の設定値をもつパラメータ
    - GATEWAYなど

- 仮定：設計者は暗黙のルールを持って設計している
  - 例）命名規則，アドレス割り振り規則など
- 各サーバ，データセンタを比較⇒設計パターンを推定可能

新たな設計パターンを特定
⇒自動化率を向上

設計パターンの推定

FUJITSU

パターン推定手順

- ①各Availability Zone (AZ) の構成情報を構築した順に並べる
- ②各AZを比較して設定値が変わる部分を検出
- ③AZの構築時期とパラメータとの比例関係を推定
  - 例）構築時期を1つ経るごとに3オクテッド目が1増える
- ④比例関係から新規AZの設計パターンを推定
  - IF “AZ3” THEN “Gateway=10.0.3.1”

過去

AZ0	サーバA	サーバB	サーバC	サーバD
IPアドレス	172.16.0.1	172.16.0.2	172.16.0.3	172.16.0.4
ゲートウェイ	10.0.0.1	10.0.0.1	10.0.0.1	10.0.0.1
AZ1	**サーバA**	**サーバB**	**サーバC**	**サーバD**
IPアドレス	172.16.1.1	172.16.1.2	172.16.1.3	172.16.1.4
ゲートウェイ	10.0.1.1	10.0.1.1	10.0.1.1	10.0.1.1
AZ2	**サーバA**	**サーバB**	**サーバC**	**サーバD**
IPアドレス	172.16.2.1	172.16.2.2	172.16.2.3	172.16.2.4
ゲートウェイ	10.0.2.1	10.0.2.1	10.0.2.1	10.0.2.1

最新

再帰的クラスタリングを用いる設計手順生成について

FUJITSU

入力（教師データ）

既存システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N

AZ_2	Server2-1	Server2-2	…	Server2-S_2

AZ_1	Server1-1	Server1-2	…	Server1-S_1
Parameter1	$V_{11}(1)$	$V_{12}(1)$	…	$V_{1S_1}(1)$
Parameter2	$V_{11}(2)$	$V_{12}(2)$	…	$V_{1S_1}(2)$
Parameter3	$V_{11}(3)$	$V_{12}(3)$	…	$V_{1S_1}(3)$
…	…	…	…	…
ParameterP_{11}	$V_{11}(P_{11})$	$V_{12}(P_{12})$	…	$V_{1S_1}(P_{1S_1})$

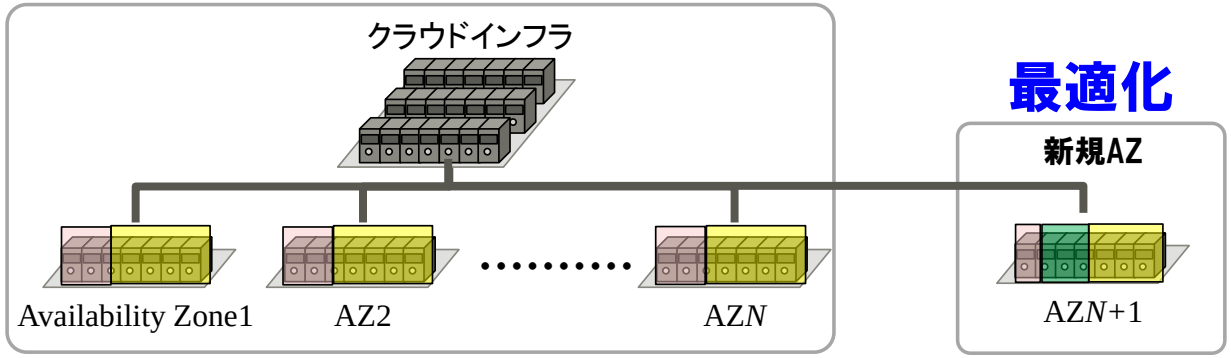
出力

新規システム構成情報

AZ_N	Server$N$-1	Server$N$-2	…	Server$N$-S_N
Parameter1	$V_{N1}(1)$	$V_{N2}(1)$	…	$V_{NS_N}(1)$
Parameter2	$V_{N1}(2)$	$V_{N2}(2)$	…	$V_{NS_N}(2)$
Parameter3	$V_{N1}(3)$	$V_{N2}(3)$	…	$V_{NS_N}(3)$
…	…	…	…	…
ParameterP_{N1}	$V_{N1}(P_{N1})$	$V_{N2}(P_{N2})$	…	$V_{NS_N}(P_{NS_N})$

提案手法

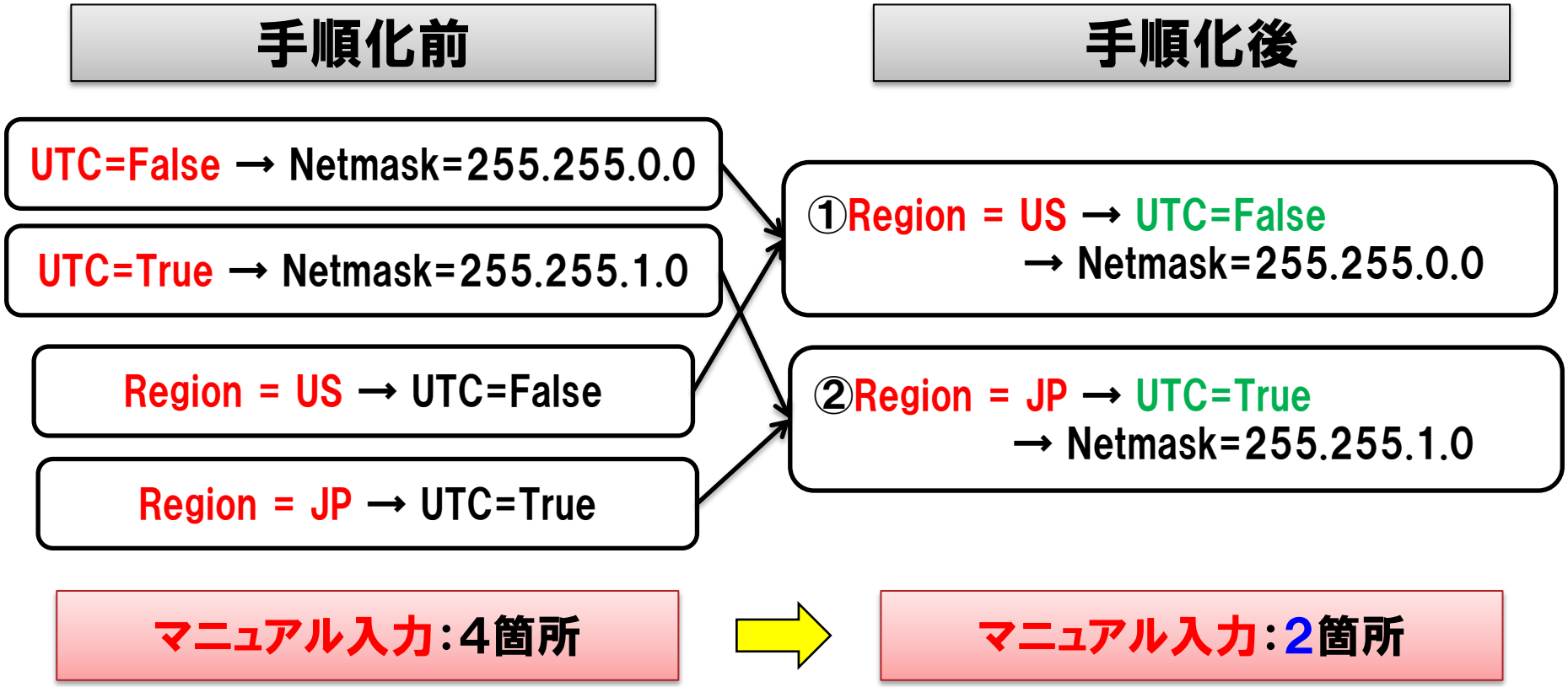
既存手法 → パラメータ推定 → 再帰的クラスタリングを用いる設計手順生成



再帰的クラスタリングを用いる設計手順生成

FUJITSU

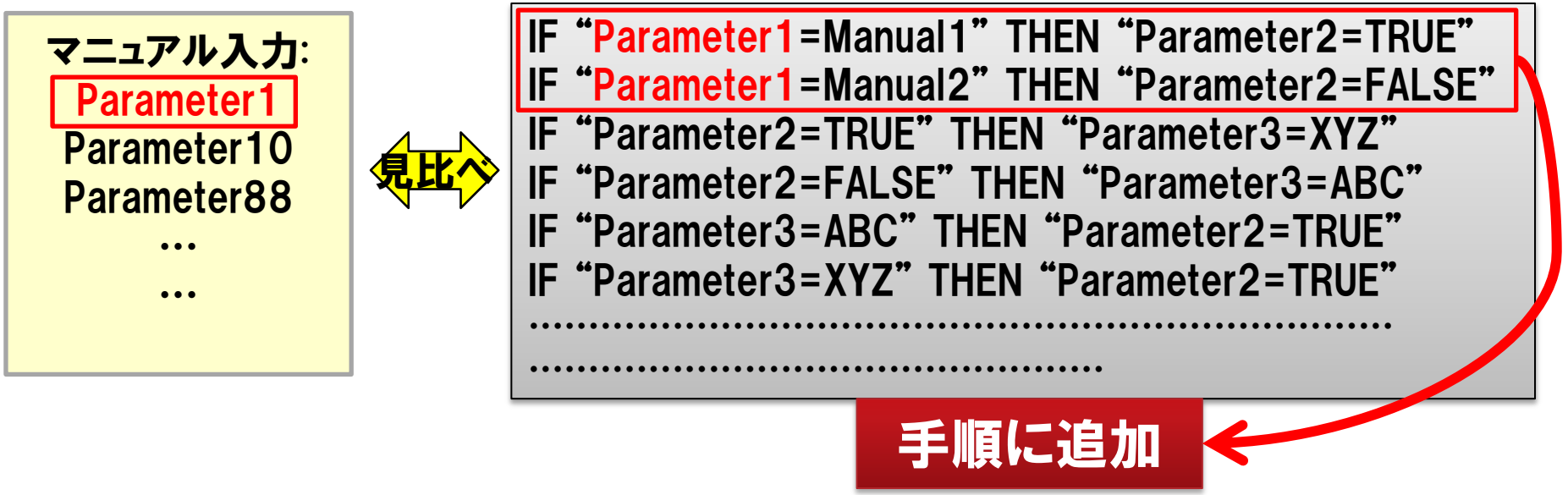
- 設計パターンを芋蔓式に成立する順番にまとめる
  - 最小限のマニュアル入力で全てのパラメータを設計可能にする
  - 例）



手順生成アルゴリズム（1）

FUJITSU

- 設計パターンを成立する順番にまとめる
  - ①決定木解析およびパラメータ推定で，設計パターンを特定できなかったパラメータをマニュアル入力パラメータとする
    - 傾向が存在しないパラメータ（グローバルIPアドレスなど）
  - ②マニュアル入力のみでIFの条件を満たす設計パターンを手順に追加



手順生成アルゴリズム (2)

FUJITSU

■設計パターンを成立する順番にまとめる

■③手順に追加したパターンのTHENの定義を既知情報とする

IF “Parameter1=Manual1” THEN “Parameter2=TRUE”
IF “Parameter1=Manual2” THEN “Parameter2=FALSE”

反映

既知情報:
“Parameter2=TRUE”
“Parameter2=FALSE”

■④マニュアル入力と既知情報でIFの条件を満たす設計パターンを手順に追加

マニュアル入力:
Parameter1
Parameter10
……

既知情報:
“Parameter2=TRUE”
“Parameter2=FALSE”

見比べ

IF “Parameter2=TRUE” THEN “Parameter3=XYZ”
IF “Parameter2=FALSE” THEN “Parameter3=ABC”
IF “Parameter3=ABC” THEN “Parameter2=TRUE”
IF “Parameter3=XYZ” THEN “Parameter2=TRUE”
……
……

手順に追加

手順生成アルゴリズム(3)

FUJITSU

- 設計パターンを成立する順番にまとめる
  - ⑤全てのパターンを手順に追加するまで繰り返す

- 例外処理
  - IFの条件を満たすパターンが存在せず，手順に追加できるパターンがない場合
  - “循環参照”を起こしている可能性がある

この時

再帰的クラスタリングアルゴリズムで解消する

循環参照とは

FUJITSU

- 互いのTHENで示す定義が互いのIFで示す条件に存在する状態

パターンA

```
IF GATEWAY="A" & LANG="B" & ZONE="C"
THEN NETMASK = "D"
```

パターンB

```
IF GATEWAY="E" & NETMASK ="D"
THEN ZONE ="C"
```

- NETMASK, ZONEは互いが互いを条件に持っている

  ⇒このままではどちらも**条件を満たすことが出来ない**

- これを**循環参照**と呼ぶ

再帰的クラスタリングアルゴリズム（1）

FUJITSU

■①循環参照が起きているパターンを検出

パターンAパターンB

IF GATEWAY=“A” & LANG=“B” & ZONE=“C”
THEN NETMASK = “D”

IF GATEWAY=“E” & NETMASK =“D”
THEN ZONE =“C”

■②循環参照しているパラメータを１つ一時除外

- ■例）パラメータ「ZONE」を入力する教師データから一時除外
- ■「ZONE」が関わらないパターンを生成できるようになる

再帰的クラスタリングアルゴリズム(2)

FUJITSU

■ ③教師データを再度クラスタリングして新パターンを得る

新パターン

```
IF  GATEWAY=“A”  &  LANG=“B”  &  nameserver=“E”
THEN  NETMASK = “D”
```

■ ④循環参照がなくなるまで繰り返し，解消した場合は除外したパラメータを元に戻す

- 例)「ZONE」を教師データに戻す

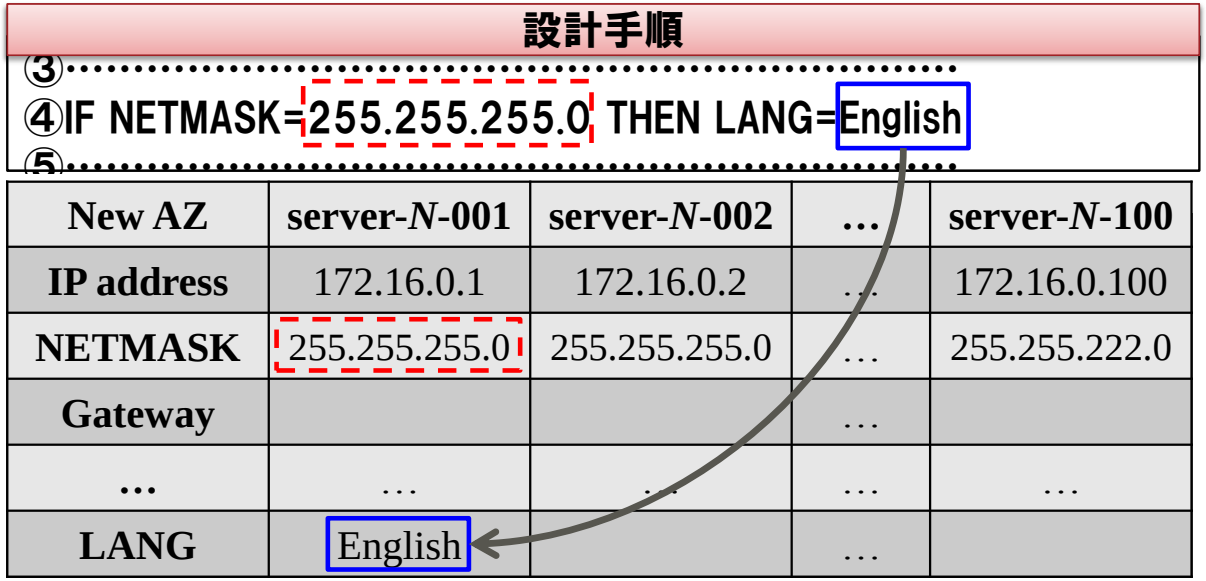
■ ※循環参照が見つからない場合

- 任意のパラメータの1つをマニュアル入力とする
- これにより全てのパラメータを力バーできる

パラメータ自動設計

FUJITSU

- 設計手順を用いてパラメータを自動設計する
  - ①設計パターンを特定できなかったパラメータの設計値を設計者に入力してもらう
  - ②設計手順の順番通りに設計パターンを適用していく
  - ③IF文を満たすサーバにTHEN文の定義を割り当てる
  - ④全ての設計パターンを適用し終えたら自動設計は終了となる



評価

FUJITSU

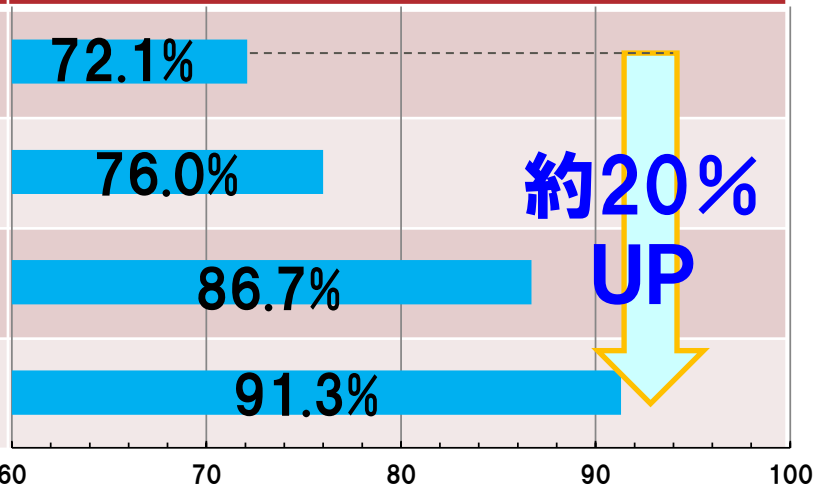
- 評価方法
  - 既存の複数AZの構成情報から新規AZのパラメータを自動設計
  - 実際の設定と比較して，どれだけ自動で設計出来たかを評価

- 入力する教師データ
  - 実際に稼働しているクラウドデータセンタから収集
  - 既存のAZの中から最新2個を選択
    - 物理サーバ：221台
    - パラメータと設定値のペア：31870個（約144種類×221台）
  - 既存AZの構成情報には誤りがないとみなし，新期AZは直近に構築したAZと設計傾向が同じであるとする

結果

- ケース1：既存技術
- ケース2：既存技術＋パラメータ推定
- ケース3：既存技術＋再帰的クラスタリング
- ケース4：提案手法全て

	パラメータ推定	再帰的クラスタリング
ケース1	×	×
ケース2	○	×
ケース3	×	○
ケース4	○	○



※自動設計割合＝自動で設計した箇所／全設計項目（12906）

考察

FUJITSU

- 既存手法と比べて自動設計割合を約20%増大することが出来た理由
  - パラメータ推定手法の効果：既存手法ではマニュアル入力としていたパラメータの設計パターンを抽出できた
  - 再帰的クラスタリングの効果：循環参照を解消し，全ての設計パターンを新規インフラの設計に適用できるようにした

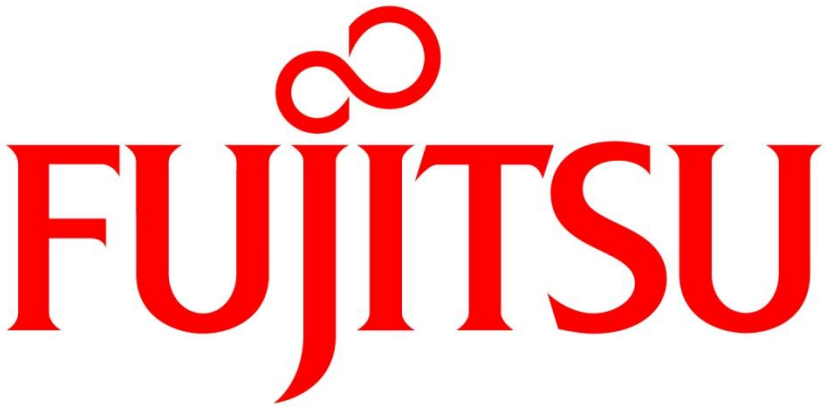
- 問題点
  - サーバのモデルチェンジなどにより，新規インフラから突然設計が変わる場合，誤った設計をしてしまう可能性がある
  - 設計がいままでと変わる際は，あらかじめ管理者によるマニュアルでの入力が必要と考えられる

まとめ

FUJITSU

- 決定木解析を用いるパラメータ自動設計手法を提案
  - 既存インフラの構築時期（世代）とパラメータの比例関係に着目
  - 世代を経るごとに等差的に設定値が変化するパラメータの設計パターンを推定
  - 決定木解析を繰り返すことで自動設計に最適な手順を生成
  - 新規クラウドインフラのパラメータ（12906箇所）の内91.3％を自動設計

- 今後
  - クラウドインフラ特有のエンハンスやアップデートを繰り返す環境に追従する取組みを行っていく予定



shaping tomorrow with you