

なぜなにZabbix

2020年 11月 13日
NTTコムソリューションズ株式会社
プラットフォームソリューション部

Transform your business, transcend expectations with our technologically advanced solutions.

日々マイナーバージョンアップ。
半年ごとにメジャーバージョンアップして
驚異的な速度で進化していく **ZABBIX**

進化した内容については
マニュアルの「What's New」で語られているけど
正直、何がどう変わったのかわからないまま
使ってる人も多いと思います。

ここでは、そんなZabbixの秘密を
みなさんと一緒に見ていきたいと思います。

『驚異の部屋』をご案内いたします

【第1の部屋】

いとおかし

Zabbixとお菓子の間には何か関係があるのか？



nabe) @_BSmile_ · 2011年12月24日
ン作った。デザインセンスなさ過ぎw



これは有名な話ですね。

Zabbix1.8.4に内包されていた、重大なバグが発祥

エラー ログイン銘菓パスワードが間違っています

ログイン



アカウント名

パスワード

Enter

Zabbix 1.8.4 Copyright 2001-2010 by SIA Zabbix

最適なICTソリューションで、
明日のビジョンをカタチにする



NTTコム ソリューションズ

【第2の部屋】

『切り捨て御免』はレバー1回転+P

扱えない精度の数値を送り込むとDBが勝手に丸める

```
value | double(16,4) |
```


※ historyテーブルの定義より抜粋

ほらね

```
[zabbix@Staging18Cent6-64bit ~]$ /usr/bin/zabbix_sender -z 127.0.0.1 -s host -k tr.numeric -o 98.7654399
Info from server: "Processed 1 Failed 0 Total 1 Seconds spent 0.000144"
sent: 1; skipped: 0; total: 1
[zabbix@Staging18Cent6-64bit ~]$ /usr/bin/zabbix_sender -z 127.0.0.1 -s host -k tr.numeric -o 98.7654399
Info from server: "Processed 1 Failed 0 Total 1 Seconds spent 0.000044"
sent: 1; skipped: 0; total: 1
[zabbix@Staging18Cent6-64bit ~]$ /usr/bin/zabbix_sender -z 127.0.0.1 -s host -k tr.numeric -o 98.7654399
Info from server: "Processed 1 Failed 0 Total 1 Seconds spent 0.000130"
sent: 1; skipped: 0; total: 1
```

タイムスタンプ	値
2020.6月.22 12:21:07	98.7654
2020.6月.22 12:20:39	98.7654
2020.6月.22 12:20:20	98.7654
2020.6月.22 12:18:12	98.7654
2020.6月.22 12:17:54	98.7654
2020.6月.22 12:17:18	98.7654

つまりこんなトリガーを作ると

<input type="checkbox"/>	深刻度	ステータス	名前 	条件式
<input type="checkbox"/>	未分類	有効	Trigger(last #3)	{host:tr.numeric.last(#3)}>98.7654321
<input type="checkbox"/>	未分類	有効	Trigger(last)	{host:tr.numeric.last(0)}>98.7654321
<input type="checkbox"/>	未分類	有効	Trigger(prev)	{host:tr.numeric.prev(0)}>98.7654321

ヒストリがコレ⇒
だから

タイムスタンプ	値
2020.6月.22 12:21:07	98.7654
2020.6月.22 12:20:39	98.7654
2020.6月.22 12:20:20	98.7654
2020.6月.22 12:18:12	98.7654
2020.6月.22 12:17:54	98.7654
2020.6月.22 12:17:18	98.7654

あ、あれ??

ステータス	最新の変更 ↓	経過時間	継続期間	障害対応済	ホスト	名前
	2020 6月 22 12:20:39	2m 19s		障害対応コメント (1)	host	Trigger(prev) {host:tr.numeric.prev(0)}>98.7654321
障害	2020 6月 22 12:20:39	2m 19s	2m 19s	未対応		
正常	2020 6月 22 12:17:54	5m 4s	2m 45s			
	2020 6月 22 12:20:20	2m 38s		障害対応コメント (1)	host	Trigger(last) {host:tr.numeric.last(0)}>98.7654321
障害	2020 6月 22 12:20:20	2m 38s	2m 38s	未対応		
正常	2020 6月 22 12:17:18	5m 40s	3m 2s			
	2020 6月 22 12:18:12	4m 46s		障害対応済	host	Trigger(last #3) {host:tr.numeric.last(#3)}>98.7654321
正常	2020 6月 22 12:18:12	4m 46s	4m 46s			



アイエエエエエ！！ ニンジャ！？ ニンジャナンデ！？

Zabbix内部では、数値はdouble型データとして扱われる



DBはdouble(16,4)なので、小数点以下第5位で四捨五入される



履歴の画面で見ると小数第4位までのデータになる

その秘密はここに

```
mysql> desc items;
```

Field	Type	Null	Key	Default	Extra
itemid	bigint(20) unsigned	NO	PRI	0	
type	int(11)	NO		0	
snmp_community	varchar(64)	NO			
snmp_oid	varchar(255)	NO			
snmp_port	int(11)	NO		161	
hostid	bigint(20) unsigned	NO	MUL	0	
description	varchar(255)	NO			
key_	varchar(255)	NO			
delay	int(11)	NO		0	
history	int(11)	NO		90	
trends	int(11)	NO		365	
lastvalue	varchar(255)	YES		NULL	
lastclock	int(11)	YES		NULL	
prevvalue	varchar(255)	YES		NULL	
status	int(11)	NO	MUL	0	

historyテーブルでは double(16,4)
itemsテーブルでは varchar(255)



なので、itemsテーブルの中では丸めが発生しない



itemsテーブルの情報を使うlast(#1)/ last(#2) / prevと
Historyテーブルの情報を使うlast(#3)では
トリガー判定の根拠となるヒストリがそもそも違う

さらに Zabbix2.2以降だと

■Itemsテーブルに lastvalue / prevvalue が無くなったので

⇒ Value Cache が使われる場合はdouble精度での判定

⇒ 一回でもDBに格納されると(16,4)に丸められる

なので、再起動やValueCacheのLowMemoryModeによって判定結果がブレる

Zabbix5.0の新機能より

Extended range of numeric (float) values

Numeric (float) data type now supports precision of approximately 15 digits and range from approximately $-1.79E+308$ to $1.79E+308$ (with exception of **PostgreSQL 11 and earlier versions**). This is by default for new installations. For upgraded installations, a **manual patch** must be applied.

5.0でこの新機能追加(というか機能改善)が行われたので
もうこんな空間の歪みは発生しない



最新バージョンのZabbix 使おうぜ！！

お客さまや社会とともに
理想と未来を追求します。



NTTコム ソリューションズ

【第3の部屋】

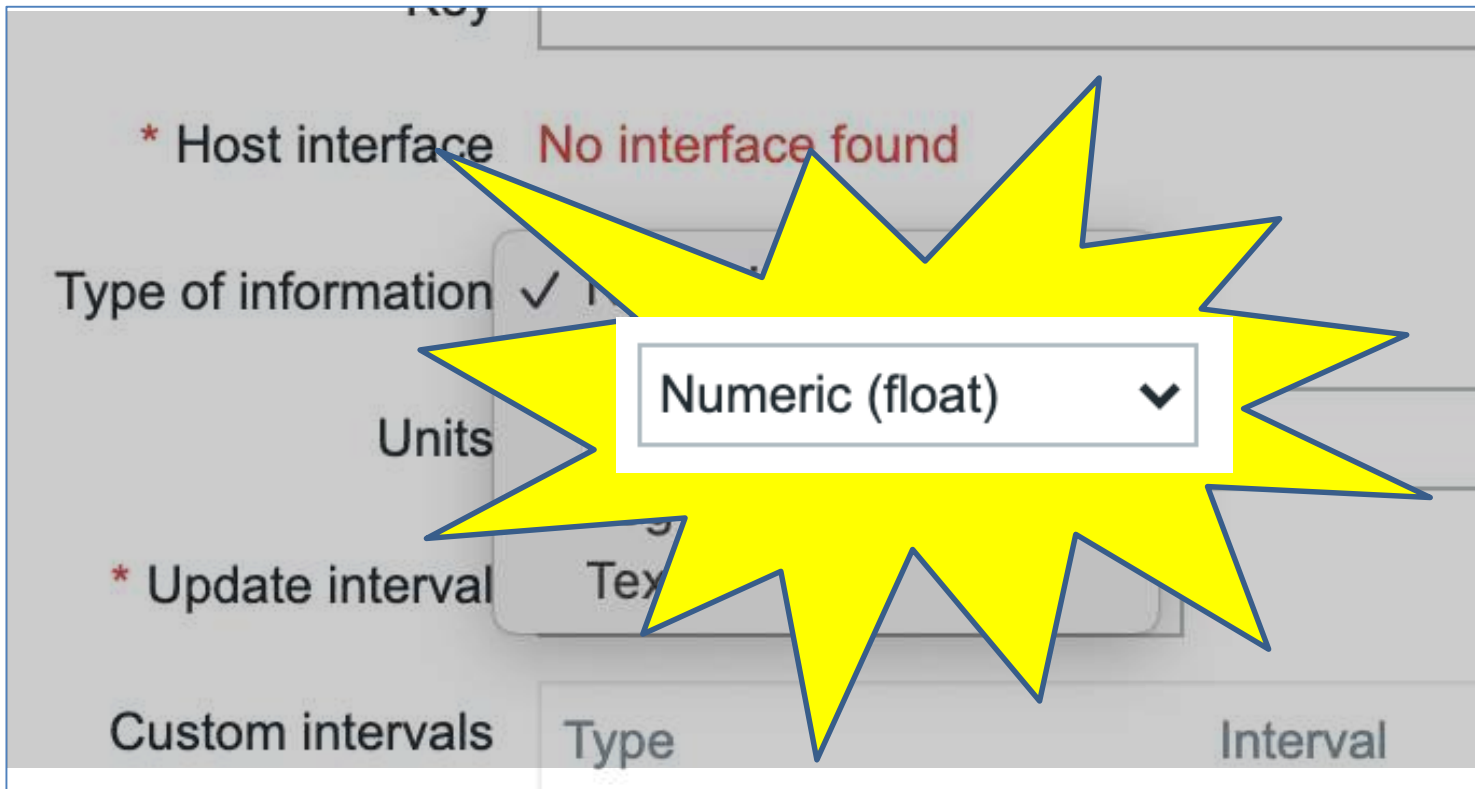
『メロンフロート
アイスをダブルで』
って言ってみたい

日本人ってUIの言語を英語にしたりしないですね。

The screenshot shows a software interface with several fields and a dropdown menu. The fields are:

- * Host interface: No interface found
- Type of information: A dropdown menu is open, showing options: Numeric (unsigned), Numeric (float), Character, Log, and Text.
- Units: A text input field.
- * Update interval: A text input field.
- Custom intervals: A table with two columns: Type and Interval.

ん! ?



昔からずっと「Numeric(float)」って書いてあったけど

- **float:** The float data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the [Floating-Point Types, Formats, and Values](#) section of the Java Language Specification. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the [java.math.BigDecimal](#) class instead. [Numbers and Strings](#) covers `BigDecimal` and other useful classes provided by the Java platform.
- **double:** The double data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the [Floating-Point Types, Formats, and Values](#) section of the Java Language Specification. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.

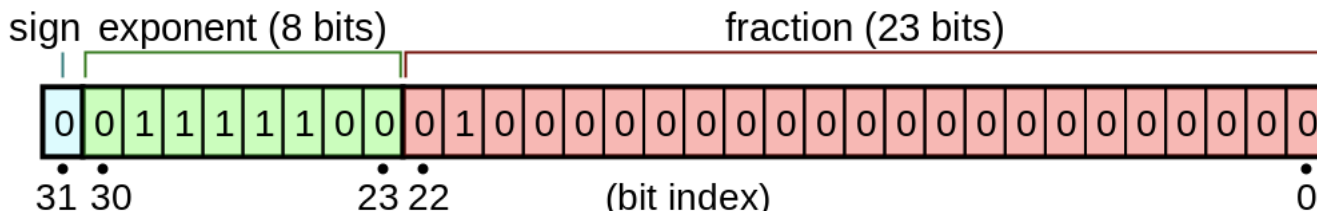
※ Oracle社 「The Java™ Tutorials」 より引用

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

3行以上の文章は読めないののでわかりやすくまとめると

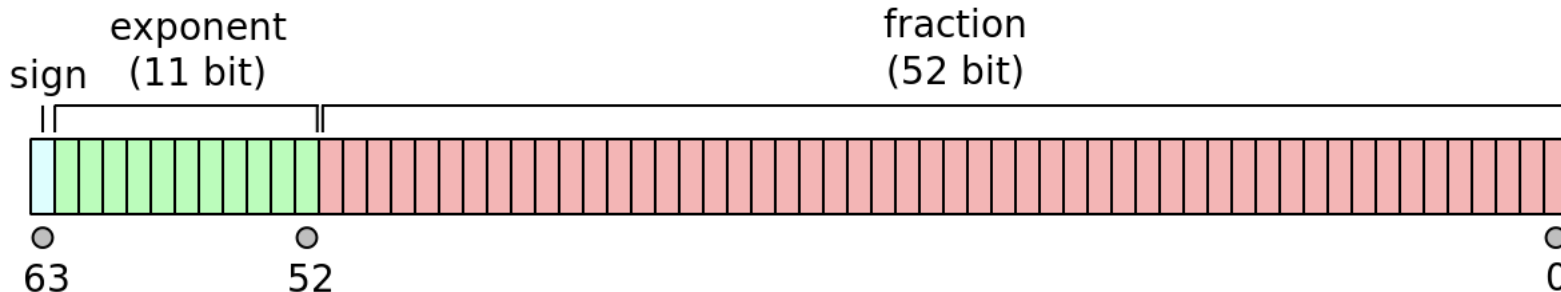
float

単精度浮動小数点数(4bytes)



double

倍精度浮動小数点数(8bytes)



float

単精度浮動小数点数(4bytes)

3.4028234

$\times 10$ の38乗

double

倍精度浮動小数点数(8bytes)

1.7976931348623157

$\times 10$ の308乗



あれれ——？
おっかしいぞ——!?

by 体は大人、頭脳は子供の名探偵。
じっちゃんの名に懸けて 真実は一つ

Zabbix5.0の新機能より

Extended range of numeric (float) values

Numeric (float) data type now supports precision of approximately 15 digits and range from approximately **-1.79E+308 to 1.79E+308** (with exception of **PostgreSQL 11 and earlier versions**). This is by default for new installations. For upgraded installations, a **manual patch** must be applied.

`net.tcp.service[ntp]` が
`net.udp.service[ntp]` に変わったように

`Numeric(float)` も `Numeric(double)` に改名しようぜ！

すべてはお客さまと社会のために



NTTコム ソリューションズ

【第4の部屋】

ダブルアップ！

俺がお前で！お前が俺で！

マイティ！マイティ！ブラザーズ！

ダブルエーックス！

Zabbixの内部処理では時刻表現を

- ・ clockという「UNIX時間(POSIX time)」
- ・ nsという「ナノ秒」

の2つを組み合わせて行っているのは有名な話

Field	Type	Null	Key	Default	Extra
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
value	double	NO		0	
ns	int(11)	NO		0	

1.8までは、時刻はclockでしか管理していなかったので
1つのアイテムが1秒以内に複数のヒストリを収集すると

{ITEM.VALUE} のマクロの値が入れ替わってしまう

問題がありました。

これも有名な話ですね

第4の部屋

itemid	clock	value
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-91
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-92
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-93
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-94
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-95
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-96
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-97
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-98
22175	1281546154	Aug 12 14:12:56 zbxs04 zabbix: error-99

2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-91	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-92	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-93	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-94	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-95	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-96	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-97	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-98	障害	致命的な障害
2010.8月.12 02:02:34	logAlarm ITEM.VALUE---Aug 12 14:12:56 zbxs04 zabbix: error-99	障害	致命的な障害

こんな大事な機能改善なのに、What's Newでの扱いはひどい

5.4.5 IMPROVED HISTORY (DB) SYNCER AND ESCALATOR PERFORMANCE

The amount of work history syncer and escalator processes have to do for escalations has been reduced, which resulted in improved performance when processing large amount of events.

5.5 OTHER IMPROVEMENTS

5.5.1 NANOSECOND SUPPORT



問題を解決するため、時刻情報の解像度を拡張し更に詳細な時刻情報としてナノ秒を導入しました。

5.5.2 DATABASE INTEGRITY

https://www.zabbix.com/documentation/2.0/manual/introduction/whatsnew200#nanosecond_support

なので代わりに説明すると

ナノ秒レベルで時刻情報を管理するために追加の時刻情報「ns」を追加することにより従来の**1秒以内**という広い範囲ではなく**1ナノ秒以内**という非常に短い期間に複数のヒストリが収集されない限りは{ITEM.VALUE}のマクロの値が入れ替わるような事象が起こらないように改善されました。

私たちは現在と未来を「つなぐ」
イノベーションパートナーです。



NTTコム ソリューションズ

【第5の部屋】

nanosecに単位をそろえると

蒸着は 50,000,000ns

赤射は 1,000,000ns

焼結は 1,000,000ns

あれ？

そもそもnanosecって
何桁まで入るんだ？





簡単だよ。11桁入るんだ。
つまり 99,999,999,999 だね。

```
MariaDB [zabbix]> desc history;
```

Field	Type	Null	Key
itemid	bigint(20) unsigned	NO	MU
clock	int(11)	NO	
value	double	NO	
ns	int(11)	NO	

4 rows in set (0.001 sec)



```
| ns | int(11)
```

じゃ『11』って何よ！？
PostgreSQLだとただの **int** じゃん



サバダバサバダバ〜



- ZEROFILLするときの桁数指定
- 丸カッコの中身を増減しても確保されるデータ領域は同じサイズ
- MySQLのintの上限と下限は
-2,147,483,648 ~ 2,147,483,647 (10桁)

つまり 99,999,999,999
なんて絶対入らないんですね



【 お手紙で聞いてみました 】

Dear Alexei,

This year's Zabbix Summit was held online.

So I am very sad that I couldn't go to Latvia.

I want to go to Latvia next year,

so please hold the Zabbix Summit 2021 offline.

Thank you and best regards,



Powered by Google翻訳

【 返事が来たよ！ 】

謹啓 深秋のみぎり、ZabbixConferenceJapan2020にご参加いただいている皆様におかれましてはいよいよご隆盛の由、大慶の至りと存じます。

さて、議論も活発な nanosec の桁数の件ですが Zabbixのソースコード内で**9桁に制限**しています。

だって、**ナノ秒は10のマイナス9乗**でしょ？

ざびっくすのえらいひと [Alexei Vladishev](#) より



新たなバリューを創造する
「イノベーションパートナー」
をめざします



NTTコム ソリューションズ

【第6の部屋】

電波時計って精確なイメージがあるけど
標準電波の速度が光の速度なので
300kmで1msくらいズレるらしいよ

Zabbixのヒストリデータには
その値が収集された時間が

『タイムスタンプ』

として付与されています。

Zabbix server: Context switches per second

タイムスタンプ	値
2020/11/08 19:32:18	446
2020/11/08 19:31:18	349
2020/11/08 19:30:18	350
2020/11/08 19:29:18	346
2020/11/08 19:28:18	370
2020/11/08 19:27:18	380
2020/11/08 19:26:18	532
2020/11/08 19:25:18	340
2020/11/08 19:24:18	341
2020/11/08 19:23:18	347
2020/11/08 19:22:18	361

コレ



あれ？

このタイムスタンプって
誰が付与してるんだ？

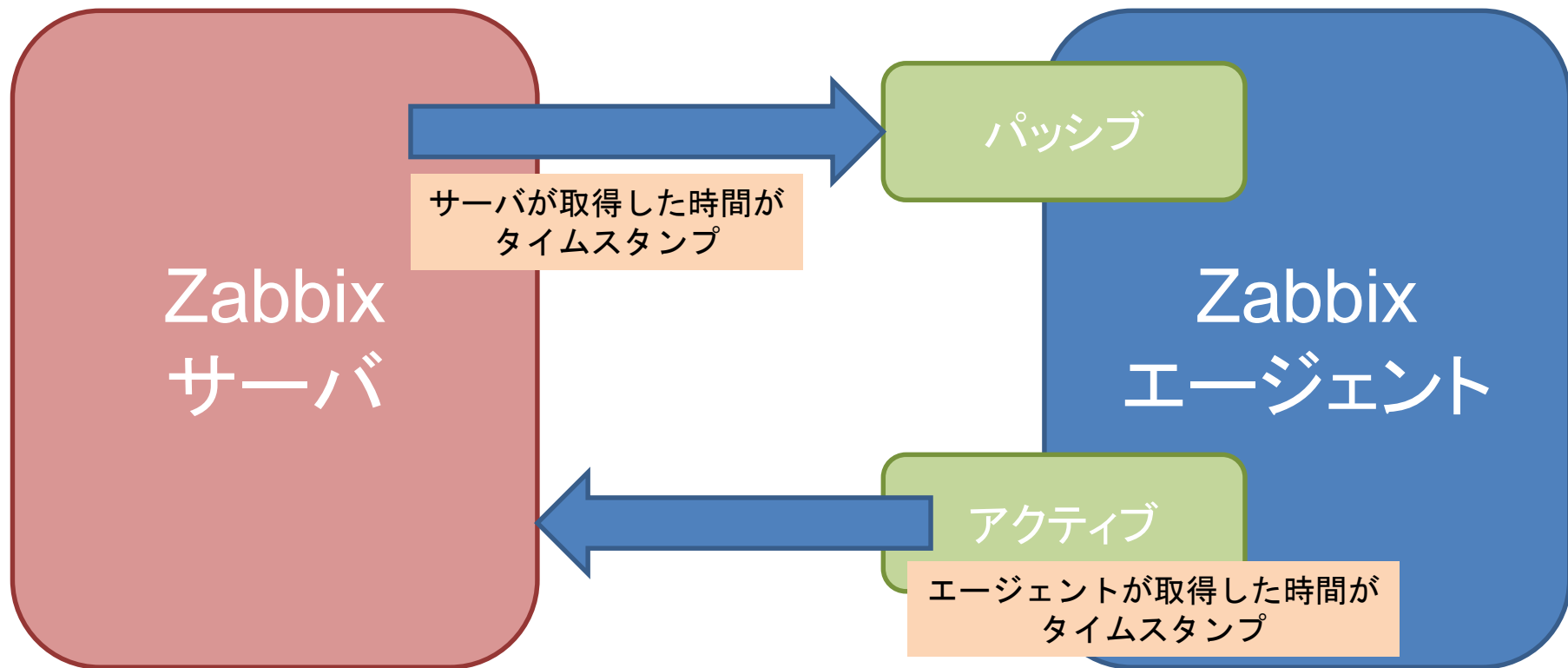


タイムスタンプを付与する人は
アイテムタイプによって違います。

- ・ Zabbixエージェント(アクティブ)
→ エージェントが付与
- ・ それ以外 (つまりポーリングタイプ)
→ 基本的にサーバが付与

※zabbix_senderとか例外はあるよ





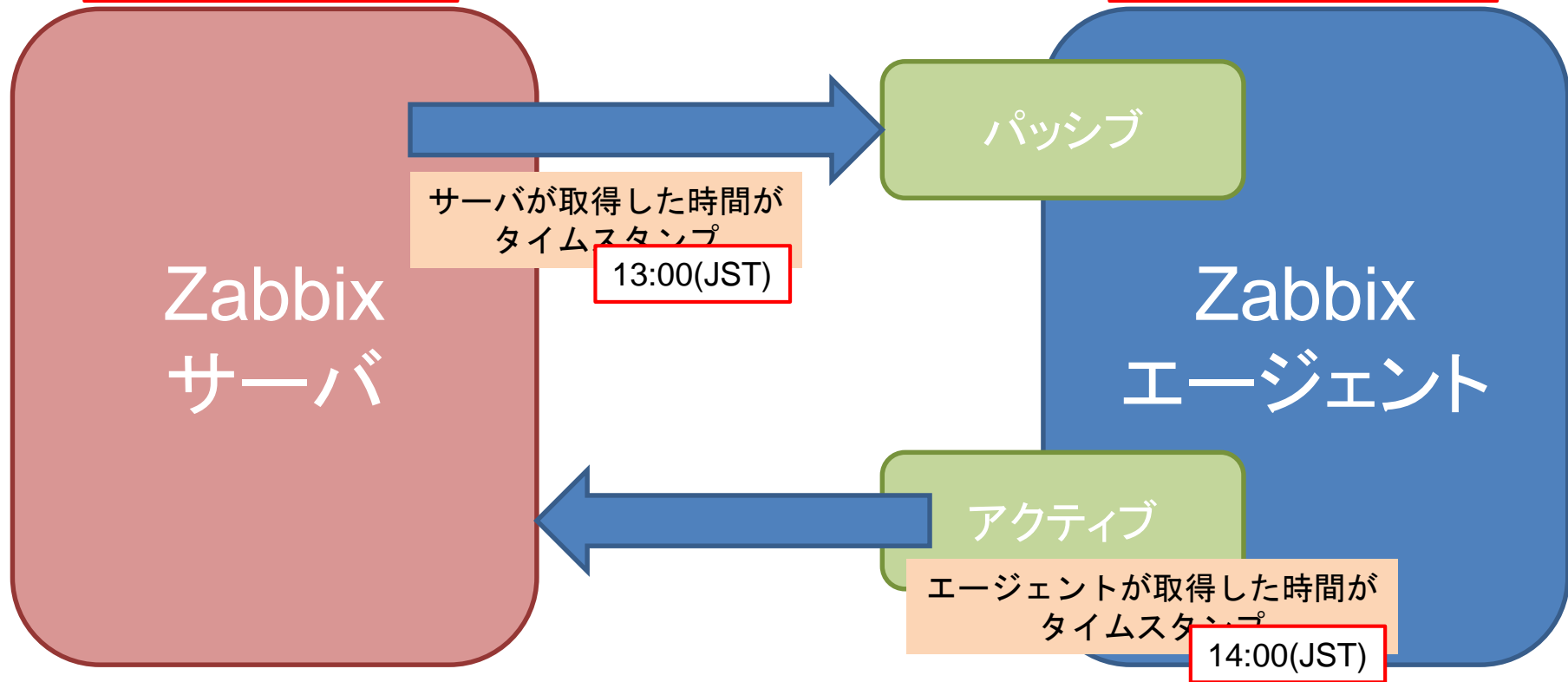
あれ？

じゃ エージェント側の
OSの時刻が狂ってたら
タイムスタンプって
ヘンな時間にならないか？



OSの時刻 : 13:00(JST)

OSの時刻 : 14:00(JST)



同じタイミングの履歴なのにタイムスタンプが違う! ?

そうならないように

- ・ Zabbixサーバ
- ・ Zabbixプロキシ
- ・ Zabbixエージェント(アクティブ)

には、時刻を補正する機能が
~~付いています。~~ いました。



OSの時刻 : 13:00(JST)

OSの時刻 : 14:00(JST)

Zabbix

14:00(JST)に送信しました。
アイテムA
タイムスタンプ 14:00
データ XXXX
アイテムB
タイムスタンプ 14:00
データ XXXX

アクティブ

エージェントが取得した時間が
タイムスタンプ

14:00(JST)

Zabbix
エージェント

OSの時刻が俺の時間と比べて+01:00だから
こいつの言ってるタイムスタンプも
きっと1時間進んでるな？
じゃ格納するときに1時間引くか

サーバの時刻を正としてヒストリのタイムスタンプは調整される

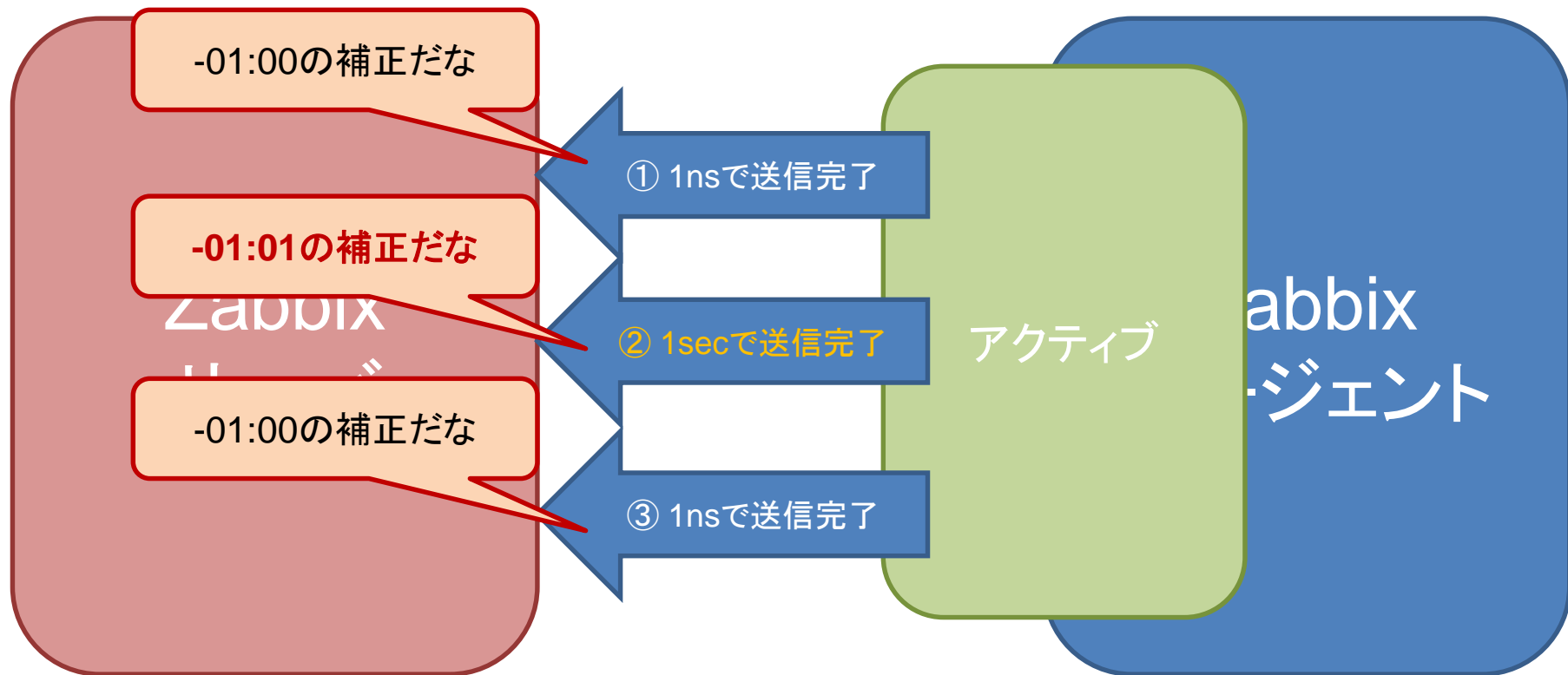
素晴らしい機能なのに、なぜかZabbix4.0で削除されました。

```
-----  
Changes for 4.0.0alpha9
```

```
Bug fixes:
```

```
.....PS. [ZBX-12957] removed time adjustment of received packets for server and proxy (MVecklers)
```

※ 『What's new in Zabbix 4.0.0』に見当たらなかったのでChangeLogから




時刻補正の影響でヒストリが入れ替わる！？

【結論】

Zabbix4.0以降を利用している場合は

**Zabbixサーバだけでなく
被監視対象もちゃんと
時刻同期をしましょう！**



これが
Chrony
らしいです



と言うか、すべてのサーバが同じ時を刻むようにするのはシステムを構成する際の基本ですよ

【お問い合わせ先】
NTTコムソリューションズ
マネジメントソリューション本部

メール: zbcm-sales@ntt.com
ZABICOM公式サイト: <https://www.zabicom.com/>



Zabbix Enterprise Applianceも扱っております

Zabbix導入のお手伝い/保守サポート/カスタマイズ開発など承ります

詳細につきましてはお問い合わせください

