intdash®

White Paper

intdash のスケーリング

2021 年 06 月 17 日

aptpod

目次

01 はじめに

このホワイトペーパーでは、双方向データ伝送プラットフォーム intdash のスケーラビリティーについて説明します。

intdash は、時系列データを高速に処理するパイプラインを提供します。5G モバイル通信の時代を迎え、世界ではますます多くのデータがやり取りされています。特に自動車や産業機械の分野では、製品の状態を正確に把握するために、製品に取り付けたセンサー、カメラ、マイクなどのデバイスを使って、秒間数千〜数万の信号や、映像、音声など多様なデータを収集し、高速に処理を行うことが増えています。しかし、高頻度かつ多様な形式のデータを漏れなく収集、伝送し、活用しやすい形式で管理することは容易ではありません。intdash は、このような大量のデータ収集・処理を可能にするプラットフォームです。

大量のデータに対応するため、intdash ではデータの特性に応じて柔軟なスケーリングが可能です。また、計測データを保存する時系列データベースについては、大規模な運用を可能にする独自のシャーディング（時系列データの分散管理）機構を備えています。

次章以降では、以下の順序でスケーラビリティーについて説明します。

- サーバーサイドの構成とスケーリングの概要 (p. 4)
- intdash サーバーの処理性能 (p. 6)
- スケーリングのポイント (p. 11)
- 時系列データベースのシャーディング (p. 13)

スケーラビリティー以外の intdash の特徴については、他のホワイトペーパー も参照してください。

02 サーバーサイドの構成とスケーリングの概要

intdash のサーバーサイドは図のような複数のコンポーネントにより構成されています。

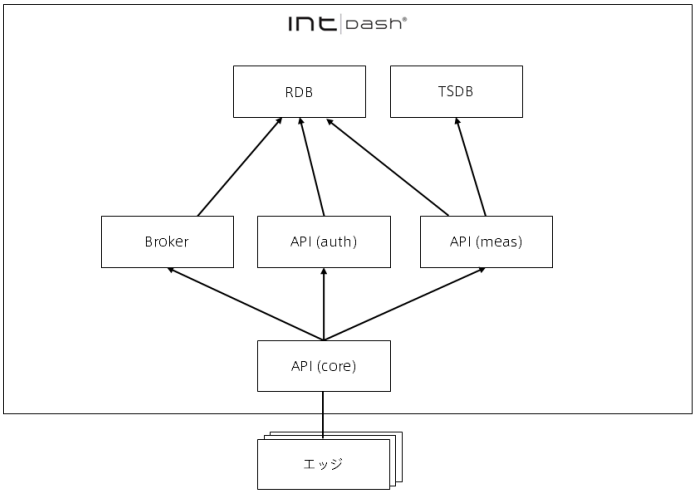

図 1　サーバーサイドのコンポーネント

それぞれの機能と、基本的なスケーリングの方法は以下の通りです。

API　API サーバーです。API サーバーでは、複数のサービスが連携して動作します。サーバーをスケールアップすることにより、性能の向上が可能です。また、ロードバランサーの配下にサーバーを複数台配置することでスケールアウトも可能です。

core
API サービスです。一部の処理については、受け取ったリクエストをそれぞれ適切なマイクロサービスに送信します。

auth
認証認可・エッジ管理に関する処理を担当するサービスです。

meas
計測に関する処理を担当するサービスです。

Broker
サーバーサイドアプリケーション間のメッセージのパブリッシュ／サブスクライブを管理し、リアルタイムデータの仲介を行います。Broker はメッセージングミドルウェア NATS を使用します。NATS サーバーはスケールアップが可能です。また、NATS クラスターのノード数を増やすことでスケールアウトすることも可能です。

TSDB
時系列データを管理する時系列データベースです。データ流量が増加して時系列データベースがボトルネックになるのを防ぐために、intdash は独自のシャーディング機構を備えています。これにより、時系列データベースのノードを増やし、柔軟にスケーリングをすることが可能になっています。詳細については、時系列データベースのシャーディング (p. 13) を参照してください。

RDB 計測の属性、エッジの属性など、時系列データ以外のデータを管理するリレーショナルデータベースです。リレーショナルデータベースのスケーリングは、レプリケーションによる読み込み負荷の分散、クラスタリング、Amazon Aurora の利用など、一般的な手法により行うことができます。

intdash のアプリケーションは複数のマイクロサービスに分かれており、データベースはそれらのマイクロサービスごとに分かれています。そのため、それぞれのデータベースを別のデータベースサーバーに分離して負荷を分散し、性能向上を図ることもできます。

03 intdash サーバーの処理性能

本章では、例として複数の intdash サーバーの構成を挙げ、各構成で処理できるデータポイント数の目安を示します。

これらの目安は、実際に当社にて各構成で intdash サーバーを構築し、シナリオに沿って性能を評価することにより得たものです。

3.1 評価のためのシナリオ

各構成では、以下のシナリオに従ってエッジからサーバーにリアルタイムデータを送信しました。

- intdash 環境はすべて AWS 上に構築する。
- 各エッジは、サーバーに 4000 フレーム／秒（4000 データポイント／秒）の CAN データを送信する。[*1]
- エッジの数を増やしていき、以下の条件をすべて満たした状態で処理可能な最大のデータ量を限界値とみなす。
  - レイテンシーが 100 ミリ秒未満。
  - エッジから送信されたデータがサーバーに保存されるまでに取りこぼしがない。
  - サーバーを構成する CPU の使用率が最大で 60% 程度。
  - その他、処理が継続できなくなるような異常がない。

> **注釈:** このシナリオはリアルタイムデータ伝送のみを対象にしています。過去データを時系列データベースから取り出して伝送する場合は、リアルタイムデータの伝送の約 5 倍の負荷が時系列データベースにかかります。
> 実際にシステムを構成する場合は、過去データの取り出しの頻度と、過去データを受信するエッジの数を考慮する必要がありますのでご注意ください。

[*1] 1 つのエッジあたりの送信データ量は、当社が取り扱うことの多い自動車の計測のユースケースをもとに設定したものです。

3.2 単体構成の場合の処理性能

単体構成の例として、API サーバーと DB サーバーの 2 つの EC2 インスタンスで構築した環境（単体構成 A〜E の 5 種類）で性能を測定しました。

なお、この構成のシステムには冗長性がないため、サービス断が許容される PoC（Proof of Concept）やデモでの使用を想定しています。

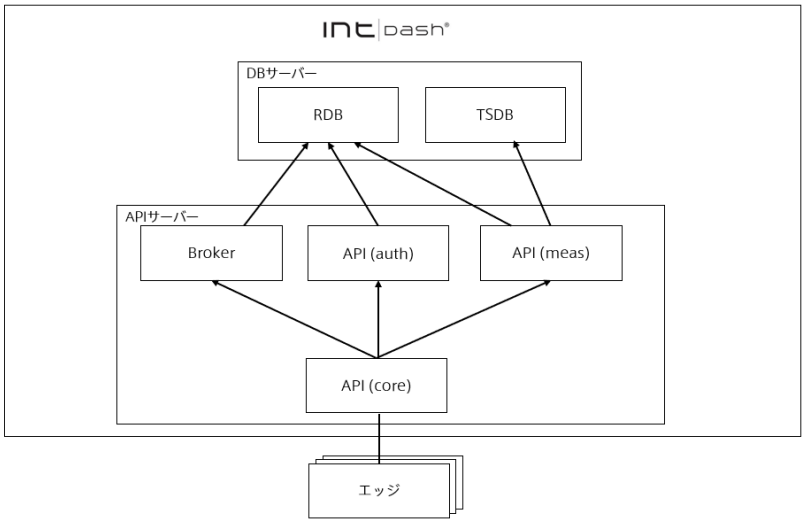

図 2　単体構成 A〜E

	API サーバー	DB サーバー	総 vCPU 数
単体構成 A	m5.large (x1)	m5.large (x1)	2
単体構成 B	m5.xlarge (x1)	m5.xlarge (x1)	4
単体構成 C	m5.2xlarge (x1)	m5.2xlarge (x1)	8
単体構成 D	m5.4xlarge (x1)	m5.4xlarge (x1)	16
単体構成 E	m5.8xlarge (x1)	m5.8xlarge (x1)	32

評価試験の結果、各構成の性能は以下のようになりました。

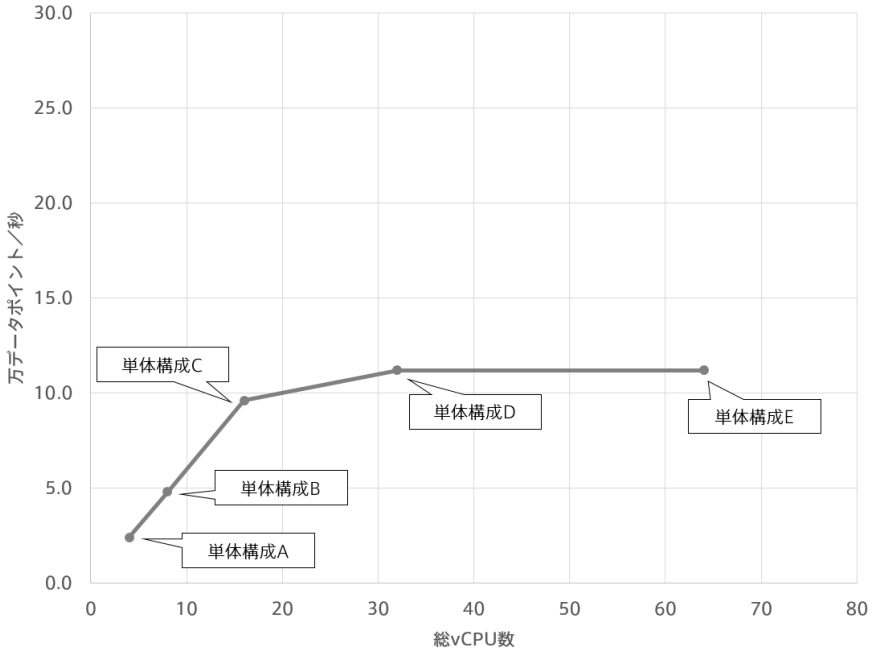

図 3　単体構成の場合の評価試験結果

インスタンスタイプをスケールアップすることによる性能向上は、データ量が 10 万データポイント／秒を超えるあたりで限界となっています（単体構成 D で限界に達しており、単体構成 E は性能が向上していません）。これは、DB サーバーにおいてストレージへの書き込みがボトルネックとなるためであることが分かっています。より多くの秒間データポイントを処理する場合には、後述のような冗長構成によるスケールアウトが必要になります。

以上の結果から、単体構成 A〜D に適したデータ量の目安をまとめると、以下のようになります。

構成	**サーバーが受信するデータポイント数（最大）の目安**
単体構成 A	2.4 万データポイント／秒
単体構成 B	4.8 万データポイント／秒
単体構成 C	9.6 万データポイント／秒
単体構成 D	11.2 万データポイント／秒

3.3 冗長構成の場合の処理性能

可用性を担保した 4 種の冗長構成（「DB のみ冗長」1 種、「全冗長構成」3 種）で性能を測定しました。

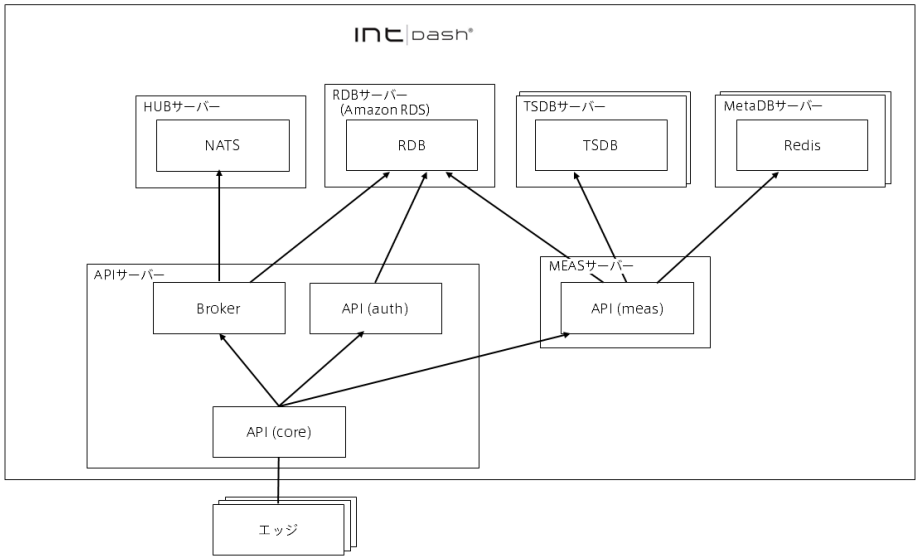

図 4　DB のみ冗長

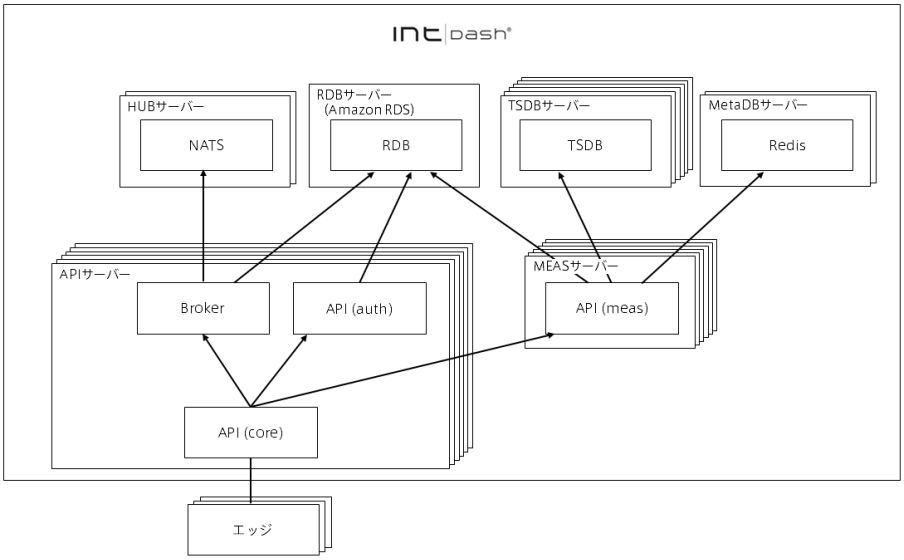

図 5　全冗長構成 A〜C

	API	MEAS	HUB	RDB	MetaDB	TSDB	総vCPU数
DBのみ冗長	m5.large (x1)	m5.large (x1)	c5.large (x1)	db.m5.large(x1)	r5.large (x2)	r5.large (x2)	16
全冗長構成A	m5.xlarge (x2)	m5.large (x2)	c5.xlarge (x2)	db.m5.xlarge(x1)	r5.large (x2)	r5.xlarge (x2)	36
全冗長構成B	m5.xlarge (x4)	m5.large (x4)	c5.xlarge (x2)	db.m5.xlarge(x1)	r5.large (x2)	r5.xlarge (x4)	56
全冗長構成C	m5.xlarge (x6)	m5.large (x6)	c5.xlarge (x2)	db.m5.xlarge(x1)	r5.large (x2)	r5.xlarge (x6)	76

- 「DB のみ冗長」は、API が冗長化されていないため、API に障害が発生するとサーバーはデータを受信できなくなります。intdash は、計測中に API が停止しても後からデータを回収する仕組みを持っているため、リアルタイムにデータを受信できなくても後からデータを回収すればよいという場合はこの構成を使用できます。
- 「全冗長構成」は、どのインスタンス 1 つが停止してもサービスの継続が可能です。ただし、API など並列処理を行っているインスタンスが停止した場合は、時間あたりの処理性能は低下します。

評価試験の結果、各構成での性能は以下のようになりました。

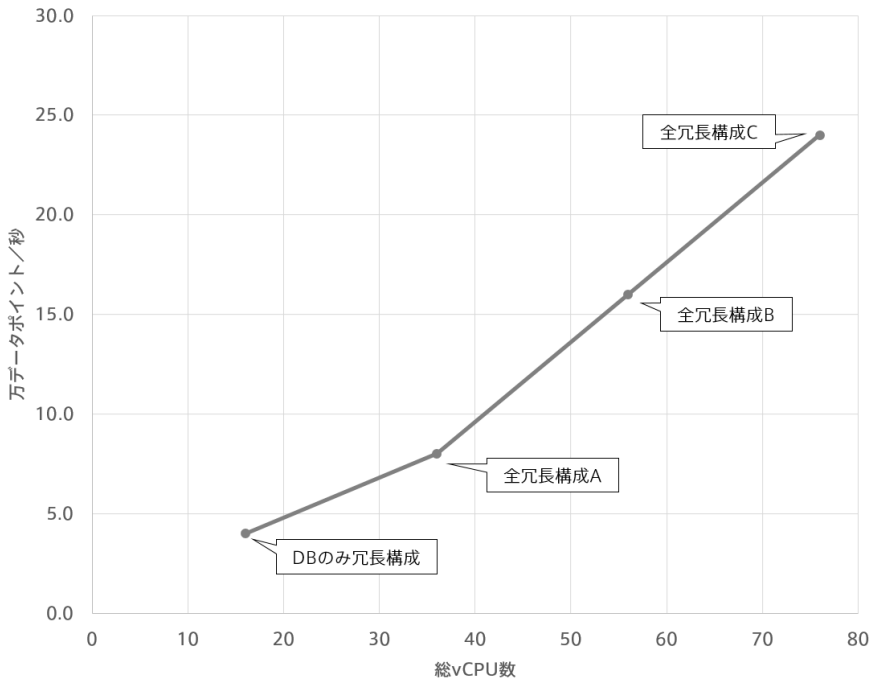

図 6　冗長構成の場合の評価試験結果

この結果から、各構成に適したデータ量の目安をまとめると、以下のようになります。

構成例	サーバーが受信するデータポイント数（最大）の目安
DB のみ冗長	4 万データポイント／秒
全冗長構成 A	8 万データポイント／秒
全冗長構成 B	16 万データポイント／秒
全冗長構成 C	24 万データポイント／秒

注釈: より高性能な構成も可能
intdash サーバーは多様な構成が可能であるため、ユースケースに合わせてさらなるスケールアップやスケールアウトも可能です。詳細についてはお問い合わせください。

なお、各構成においてボトルネックとなる個所は、データの量や種類によって異なります。ボトルネックになりやすい個所とその増強方法については スケーリングのポイント (p. 11) を参照してください。

04 スケーリングのポイント

ユースケースによって、データポイントの数、各データポイントのサイズ、送受信を行うエッジの数などの要件はさまざまです。intdash は、それらの要件に応じて柔軟な構成で使用するすることができます。

4.1 負荷の要因とそれに応じたインフラ設計

負荷に見合った適切なインフラを設計するために、負荷要因とボトルネックが発生しやすい個所を以下の表にまとめました。使い方によってどのサーバーリソースを増強するのがよいかを検討する際にこの表をご活用ください。

表の左端には負荷要因を列挙しています。それぞれの負荷要因が大きい場合に、ボトルネックになりやすいサーバーリソースを「✓✓」や「✓」で示しています。サーバーの名前は、冗長構成の場合の処理性能 (p. 9) で説明した構成例に対応しています。

例えば、アップストリームの秒間データ量が多い場合は、API のメモリー、MEAS のメモリー、TSDB のメモリーとディスク IO がボトルネックになりやすく、その部分を増強すると効果が上がる可能性があります。

負荷要因 ＼ ボトルネックになりやすいサーバーリソース		API		HUB		MEAS	
		CPU	メモリー容量	CPU	メモリー容量	CPU	メモリー容量
アップストリーム*1	秒間ユニット数の多さ（頻度）	✓				✓	
	秒間データ量の多さ（データの大きさ）	✓	✓✓			✓	✓✓
	送信エッジの多さ	✓				✓	
リアルタイムデータのダウンストリーム*2	受信エッジの多さ	✓		✓✓			
	秒間データ量の多さ（データの大きさ）	✓✓	✓	✓✓	✓		
過去データのダウンロード	受信エッジの多さ	✓				✓	
	データ量の多さ	✓	✓✓			✓	✓✓

負荷要因 ＼ スケール対象		RDB		TSDB		
		CPU	メモリー容量	CPU	メモリー容量	ディスクIO
アップストリーム*1	秒間ユニット数の多さ（頻度）	✓		✓✓		
	秒間データ量の多さ（データの大きさ）			✓	✓✓	✓✓
	送信エッジの多さ	✓✓		✓		
リアルタイムデータのダウンストリーム*2	受信エッジの多さ	✓✓				
	秒間データ量の多さ（データの大きさ）					
過去データのダウンロード	受信エッジの多さ	✓✓		✓		
	データ量の多さ			✓	✓✓	✓✓

凡例： ✓✓ スケール対象の増強による改善効果が大きい
✓ スケール対象の増強による改善効果が中程度

*1: サーバーからエッジへの送信
*2: エッジからサーバーへの送信

注釈: 時系列データベースのシャーディング用データベース MetaDB はボトルネックにはなりにくいため、上の表では省略しました。

上の表のうち、主な負荷要因とスケーリング対象の関係を以下にて説明します。

4.1.1 リアルタイムデータのアップストリームを行う場合の負荷要因

リアルタイムデータのアップストリームを行うと、API と RDB に負荷がかかります。また、計測データをサーバーに保存する場合は、MEAS と TSDB にも負荷がかかります。

- サーバーに保存する秒間データポイント数が多い場合、TSDB において CPU 性能が必要になります。
- サーバーに保存する秒間データ量が多い場合、TSDB において多くの書き込みが発生するため、ディスク IO の性能が必要になります。
- 同時にデータを送信するエッジが多い場合、同時に処理しなければならないデータポイントの数が増大するため、また属性情報が増大するため RDB の CPU 性能が必要になります。

4.1.2 リアルタイムデータのダウンストリームを行う場合の負荷要因

リアルタイムデータのダウンストリームを行うと、API と HUB に負荷がかかります。

- 同時にデータを受信するエッジが多い場合、HUB と RDB において CPU 性能が必要になります。
- データ量が多い場合、HUB において CPU 性能とメモリー容量が必要になります。

4.1.3 エッジが過去データをダウンロードする場合の負荷要因

エッジが過去データをダウンロードすると、API、MEAS、TSDB に負荷がかかります。

- 同時にデータを受信するエッジが多い場合、RDB において CPU 性能が必要になります。
- データ量が多い場合、TSDB において多くの読み出しが発生するため、ディスク IO の性能が必要になります。

4.2 スケールアップ／スケールアウトの限界

上に述べたポイントに沿ってスケーリングを行うことにより、効果的に性能を上げることができますが、いくつかの限界が分かっています。

- TSDB をスケールアップする際は、時系列データベース単体で有効に使用できる最大の CPU 性能とメモリー容量が限界点となります。また、時系列データベースのディスク IO が限界となるケースがあります。時系列データベースのスケールアウトについては、時系列データベースのシャーディング (p. 13) を参照してください。
- RDB をスケールアップする際は、リレーショナルデータベース製品の単体での性能の限界が限界点となります。なお、リレーショナルデータベースについては、スケーラブルなマネージドサービスがいくつも存在するため、intdash 独自のスケールアウトの仕組みは用意していません。
- HUB をスケールアウトする際は、NATS の性能の限界により、ダウンストリームの秒間データポイントが一定数を超えると効果が得られません。目安として、おおよそ 38 万データポイント／秒が限界となります。

05 時系列データベースのシャーディング

データの流量が増えた場合、時系列データベースはボトルネックになりやすい部分です。intdash では、OSS を組み合わせた独自のシャーディング機構により、時系列データベースの負荷を分散させています。

5.1 intdash の時系列データ

intdash では、時系列データを「計測」という単位で管理しています。1 つの「計測」とは、計測開始時刻から終了時刻までの間に特定のエッジで取得されたデータを指します。計測は一意の識別子「計測 ID」を持ちます。

例えば、典型的な自動車の計測では、自動車に設置されたエッジデバイス（intdash サーバーにデータを送信する車載コンピューター）の電源を入れることにより計測が開始され、電源を切ることにより計測が終了します。このひと続きの時系列データを 1 つの計測とします。途中、電波状況が悪くなり、リアルタイムにサーバーにデータが送信されなかった場合も、電波状況が改善した後にデータが回収され、計測開始から計測終了までが 1 つの計測として管理されます。

5.2 シャーディングによる負荷分散

1 つの計測のデータを複数のチャンクに分割して、複数の時系列データベースノードに別々に保存することにより、負荷を分散します（シャーディング）。データの分割方法は、intdash で扱うデータの特性や、intdash の読み書き処理の特性に沿って設計されています。これにより、intdash は高速性を保ったまま柔軟にスケーリングすることが可能となっています。

> **注釈:** このシャーディング機構は、時系列データベースの可用性を向上させるものではありません。可用性向上は、クラスタリングソフトウェアを使ってレプリケーションすることにより行っています。

時系列データは、永続化の際にチャンクに分割されます。このとき、「計測 ID」と「時間範囲」の組み合わせを各チャンクのキー（シャードキー）とします。

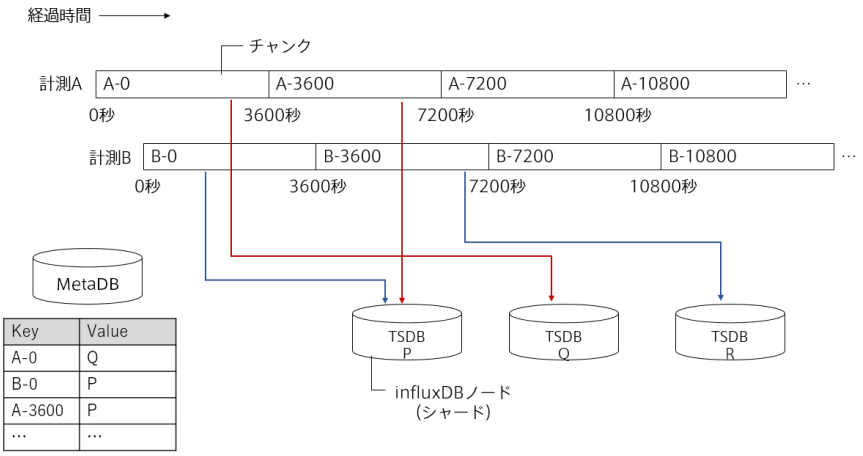

図 7　シャーディング

例えば、「計測 ID が A である計測の、開始 0 秒から」という情報が 1 つのシャードキーになります。書き込み先の時系列データベースノードは、「シャードキーのダイジェスト」と「時系列データベースノードの数」の剰余演算により決定されます。これにより、別の計測は別のシャードに振り分けられ、別の時間範囲のデータは別のシャードに振り分けられることで、負荷が分散されます。なお、時間範囲の長さは調整が可能で、計測データの時間的な長さや時間あたりのデータ量に応じて変更することができます。

時系列データベースは複数のアプリケーションから使用されますので、同じ条件であれば同じノードが選択されなければなりません。上記のようなシャードキーによる方法であれば、ロックやサーバー間の合意なしにノードを選択することができます。

時間範囲の長さ、時系列データベースノードの情報など、シャーディングに関する情報は、シャーディング情報専用のデータベース（MetaDB）に格納されます。また、シャードキーに対するノードの関連付けも、一度計算されるとシャーディング情報データベースに保存されます。そのため、ノード決定要素の 1 つである「時系列データベースノードの数」が後で変わったとしても問題はありません。

注釈: 時系列データベースのノード数の増減にも対応しています。

- ノードを増やす際には、既存のチャンクを新しいノードに配置しなおします（リバランス）。そのため、ノードを増やした場合には、ノード増加後に新たに保存されたデータだけでなく、既存のデータの読み書き性能も向上します。
- ノードを減らす際には、データが失われないよう、退役するノードのデータを他のノードに適切に配置しなおすことができます。