



## DB2いろはがるた



第3回「は」 - ハッシュング、データ分けるならこの方法



執筆者

春野 さくら

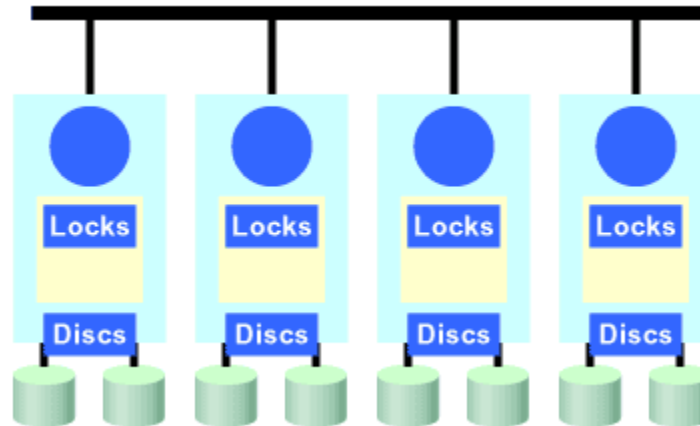
「DB2いろはがるた」を執筆するために参上した、なぞの女性。日本の古典文学を愛する。

桜の花もあつという間に満開になって散ってしまいました。春風に誘われ、夜桜見物にでかけて雅な幽玄の世界にひたると、データベースのような、やぼなものさきれいさっぱり心から消えてしまい、しばしの幸せな気分！しかし、お気楽なことばかりもいっておれず(現実は厳しい！)、この連載もはや3回目です。今回は は です。

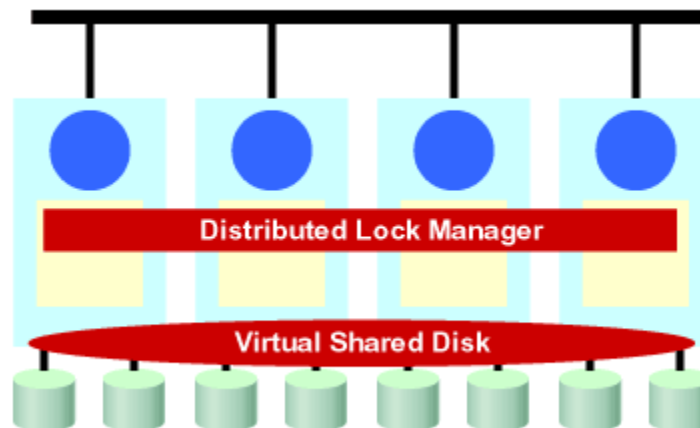
DB2 ファミリー製品には俗に、パラレル・サーバーと呼ばれる製品が2種類あります。SYSPLEX上で動くDB2/MVSと DB2 UDB エンタープライズ拡張エディションです。今回はこのDB2 UDB エンタープライズ拡張エディション(EEEと呼ぶ)のお話です。

エンタープライズ拡張エディションは、複数の物理マシン上にあるデータをあたかも一つのデータベースのように扱える製品です。OracleではOracle パラレル・サーバーにあたります。といっても、製品のアーキテクチャーや内部のしくみについてはだいぶ違います。DB2 UDB EEEは、AIX,Solaris,HP-UX, LINUX、WindowsNT,NUMA-Q上で動きますが、複数の物理マシンといってもそれらは一種類のオペレーティング・システムで、混在した環境では動きません。パフォーマンスという観点からはできるだけ、同じ処理能力のマシンでシステムを構成します。アーキテクチャー的にはDB2 UDB EEEはシェアード・ナッシングです。(これは、各々の物理マシンがリソースを共有しないという意味です。なんかあっさりした関係ですね。でもお隣のことを意識なくていいという点では都会的です。)

それに対して、Oracleのアーキテクチャーはシェアード・エプリシングです。(各ノードがデータ(ディスク)を共有します。)どちらのアーキテクチャーが優れているかについてはいろいろ論争がありましたが、技術的には、どちらも優位/不利な点があります。シェアード・ナッシングはデータをノード間で共有しないので、MPP(RS/6000-SP2のようなMassive Parallel Processors)ハードウェア・アーキテクチャーとより良くマッチします。データが増えても、物理マシンを追加することによって簡単にパフォーマンスの向上が期待できます。UNIX/Windows系ではOracleを除くすべてのRDBMSベンダーで採用されています。ただし、もし一つのノードがダウンするとその影響がデータベース全体に波及することがあるので、通常はクラスター製品とくみあわせて可用性を保証していません。



シェアード・エプリシングはディスク共有をソフトウェアでエミュレーションします。データを複数のノード(CPUとメモリー)で共有するために、データの整合性を確保するために分散ロック・マネージャが必要となります。そこで、ノードが増えれば増えるほど、データ整合性のため、ノード間のメッセージのやりとりが大量に起こり、それがボトルネックとなって、物理マシンを増やしても、十分なスケーラビリティがでないということが起こります。ただし、ディスクを共有しているので、一つのノードがダウンしてもデータベース全部には波及しません。

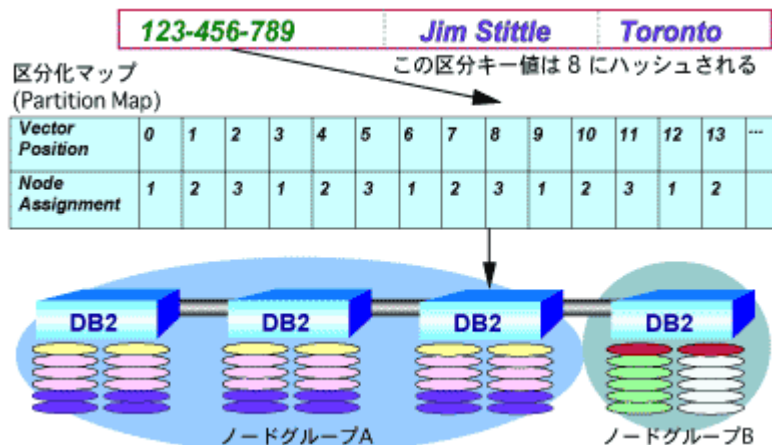


アーキテクチャーの違いの他に、データベースのデータの分け方にも種類があります。パラレル・データベースはアプリケーションからみると普通のデータベースと全く同じAPIを使うので、パラレルだからといってユーザーは何も意識する必要はありません。データベース管理システムが自動的に一つのSQL文の処理を、複数のマシンにわけて処理することによりパフォーマンスが向上するのです。(つまり司令塔が一人いて、他のチーム・メンバーに仕事を割り振って、監督しているようなものです。)そのため、データベース管理システムはあらかじめ、一つのデータベース、すなわち一つの表を分散して複数の物理マシン上にわけて構築しておく必要があります。一つの表のデータ(各行)は各ノードに分散され格納されます。各ノードのデータはそれぞれのノードのリソース(CPU&Memory)を使って処理されます。

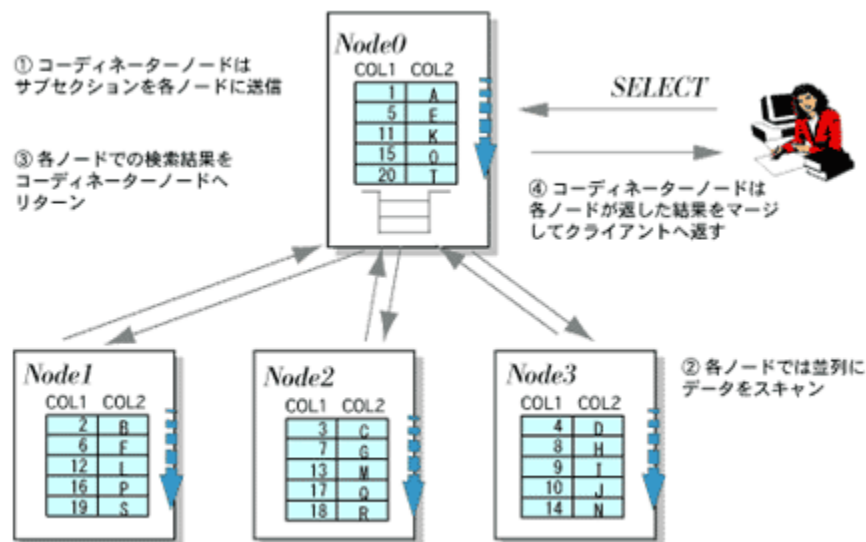
DB2 UDB EEEは内部ハッシュ関数を使って、データを各ノードに分けています。表の作成時に、表の中の特定の列をパーティション・キーとして定義する

と、行をINSERTする時に、パーティション・キーの値をハッシュングによって計算し、4096のインデックスを持つパーティション・マップにしたがって、それに対応するノード上に挿入してゆきます。

データの分け方としてはキーレンジ方式というのがありますが、DB2 UDBではその方法はとっていません。なぜなら、データの配置がノードごとに偏る危険性があり、パフォーマンスという観点からはハッシュングの方が優れているからです。ただし、キーレンジ方式にも当然業務によってはむいているものもあります。



例えば、SELECT処理時に、全表スキャンの場合は全ノードで平行処理を行い、もしWHERE文節でマッチするデータが存在するノードが判明するなら、そのノード上のみでSELECT処理を行います。DELETE、UPDATEも全く同じように平行処理を行います。DB管理システムは、本当に各物理マシンを効率的に働かせる司令官のようなものです。かっこいいですね！



[日本IBMについて](#) | [プライバシー](#) | [お問い合わせ](#)