Transactional Partitioning:
A New Abstraction for Main-Memory Databases

Vivek Shah, Marcos Vaz Salles
DMS Lab, Department of Computer Science (DIKU)
University of Copenhagen
HIPERFIT workshop, Skodsborg
Online Transaction Processing (OLTP) Application Goldmine

Reservation system
Banking system
Order Entry system
Online Transaction Processing (OLTP) Application Trends
OLTP Application Evolution

- Internet
- Computer Hardware
- Computer Software
  - Open Source
- Cloud Computing
OLTP Application Trends

Throughput

==

Latency

Resource Utilization
OLTP Application Trends

TIME TO MARKET

Under Development

Development

= Maintenance

Costs

Benefits
PVC Triangle

Motivation:

PVC != CAP

Performance Matters
Outline

- Motivation
- PVC Problem in Existing Solutions
- Logical Partition Solution
- Challenges
- Conclusion
Online Transaction Processing Application (OLTP) Properties

- Interactive
- Update heavy
- Update consistency

Highly Interactive Commodity hardware Consistent on Update HICCUP

Commodity hardware
OLTP Application Properties

Short

Fast

locality
THE OLTP System for OLTP Applications
OLTP Programming Models

- Logical Programming model
  - Not Data model
  - Abstraction of the physical layout

- Physical Implementation

Current OLTP Programming Models

- Unified view of shared state
  - Classic relational model

- Distributed storage oriented partitioned by key
  - Key-value store model
Unified View of Shared State Model

- Strong consistency, high level data model
- Hiding partitioning $\rightarrow$ Hard to reason about performance
- Partitioning is key $\rightarrow$ Houdini systems

$PVC$ or $PVC$ or $PVC$
Distributed Storage Oriented Partitioned by Key Model

- Weak/no consistency, low level storage oriented data model
- Hard to reason about locality
- Exposing partitioning → Reason about performance
- Control performance (build yourself), variety
- PVC or PVC or PVC
THE PROBLEM

How to build an OLTP system that

- Maintains ACID guarantees
- Exposes partitioning in the programming model
- Exposes program costs
- Maps the programming model to the commodity-hardware cluster
- Guarantees high resource utilization

Programming Model

Write good programs

Implementation

Run programs efficiently
Outline

- Motivation
- PVC Problem in Existing Solutions
- Logical Partition Solution
- Challenges
- Conclusion
Our Solution (Logical Partitioning)

Logical Partition = Logical unit of execution and associated storage (e.g., warehouse in TPC-C)
Accessible through function calls → Transactions
Transactions are local, invoked with logical partition
Transactions can invoke other transactions

```plaintext
txn T1 (...) {
    ...
    return res;
}
```

EXEC T1 (x_input) ON PARTITION (L0)

Invoke txn with input parameters and logical partition identifier

```plaintext
txn T2 (...) {
    ...
    input' = f(..);
    res = EXEC T1(input') ON PARTITION (L1);
}
```

Txn Invocation
Our Solution (Logical Partitioning)

txn T1 (input) {
    ......
    ......
    return res;
}

PARTITIONING FUNCTION map(input) {
    ......
    return logical_id;
}

T1 PARTITION MAPPER map;

EXEC T1 (input) on PARTITION L0;

EXEC T1(input) on PARTITION (input);
New Order

Skodsborg

Order:
2 Longjing tea - Beijing
3 Liquorice - Copenhagen

Stock:
- Longjing tea - 5
- Liquorice - 3

Item:
- Longjing tea - $4
- Liquorice - $3

Total cost: $17

Beijing

Stock:
- Longjing tea - 5

Copenhagen

Stock:
- Liquorice - 3
txn new_order (w_id, d_id, c_id, order) {
  <wh,dist,cust> = gen_order_id(w_id, d_id, c_id, order);

  total = 0;
  for(ord_item in order.items) {
    amount = get_amount(ord_item);
    total += amount;
    update_stock(ord_item, amount);
    stock_info = get_dist_info_stock(ord_item);
    add("order_line", dist.order_id, w_id, d_id, stock_info, amount, ...);
  }

  total_pay = (1 + wh.tax + dist.tax)*total* (1 - cust.discount);
  return total_pay;
}
New Order (How to use the new model?)

- Element of distribution
- Affinity of programs
- Increase in data and compute
- Warehouses (Intuitively from application)
New Order (How to use the new model?)

```
txn new_order(w_id, d_id, c_id, order) {

    <wh,dist,cust> = gen_order_id(w_id, d_id, c_id, order);

    total = 0;
    for(ord_item in order.items) {
        amount = get_amount(ord_item);
        total += amount;

        update_stock(ord_item, amount);
        stock_info = get_dist_info_stock(ord_item);

        add("order_line", dist.order_id, w_id, d_id, stock_info, amount,...);
    }

    total_pay = (1 + wh.tax + dist.tax)*total*(1 - cust.discount);
    return total_pay;
}
```
New Order Stock Update using Logical Partitioning

txn new_order_update_stock(order) {
    Result = <>;

    for(ord_item in order.items) {
        amount = get_amount(ord_item);  // Compute order item cost
        update_stock(ord_item, amount);  // Update stock
        stock_info = get_dist_info_stock(ord_item);
        append(result, <stock_info, amount>);  // Gather stock information for order line
    }

    return result;
}

PARTITIONING FUNCTION map(w_id) { return w_id; };
new_order PARTITION MAPPER map;
new_order_update_stock PARTITION MAPPER map;
txn new_order (w_id, d_id, c_id, order) {
    <wh,dist,cust> = gen_order_id(w_id, d_id, c_id, order);
    results = <>;

    for(s_id in order.supplier_w_id) {
        temp_res = EXEC new_order_update_stock
            (subset(order, s_id)) ON PARTITION (s_id);
        append(results,temp_res);
    }

    total = 0;
    for(result in results) {
        for(item_result in result) {
            total += item_result.amount;
            add("order_line", dist.order_id, w_id, d_id, item_result, ...);
        }
    }

    total_pay = (1+wh.tax+dist.tax)*total*(1-cust.discount);
    return total_pay;
}
What has changed?

- Exposed partitioning
  - Cost of communication
  - Cost of co-ordination
  - Performance is visible, controllable

- Maintained ACID
  - Isolation is good
  - No need to reason about inter-leavings
Logical Partitioning Model

- Split the programming model into logical units of storage and execution

- Application Developer does splitting → WYSWYG

- Maintain ACID guarantees

- Transactions → Code Isolation → Partitioning Element

- Programs → Produce Data

- Separation of concerns → Honesty about cleverness → One man does not fix all
Challenges

- Implementation (ongoing work)
  - Mapping logical to physical partitions
    - Reuse main-memory shared-everything engine (Silo)
    - Cost model, workload variance, skew, scheduling
  - Local Concurrency Control & Global Commit
    - Optimistic concurrency control → Global commit
    - Less is more
- Evaluation
  - TPC-C (Varied configurations of physical partitions, workload parameters)
  - Oltpbench ?
- Cloud Integration
  - Programs, performance requirements, resources
Conclusion

- Performance, Variance, Cost (PVC) → OLTP Trends
- Existing programming models do not meet PVC goals
- Logical Programming model
  - Expose partitioning → Use transactions
  - Provide global ACID guarantees
- Write Good Programs → Good abstraction
- Run good programs efficiently → Resource Utilization
- Logical Partitioning → PVC Goals → GET Ph.D.