solidDB™

IBM solidDB 6.3

Paul Chang
Senior I/T Specialist

Agenda

- Overview
- Packaging
- What's new in 6.3?
  - solidDB Universal Cache
    - Architectural overview
    - Typical In-Memory Caching examples
    - Handling Failures in non-HA Configurations
    - Universal Cache & High Availability
    - Replication Features
    - Security and Authentication
    - Restrictions
  - solidDB
    - Performance Improvements
    - UTF-8 Support
    - New configuration parameter defaults
    - Multiple secondaries support
- solidDB Techie Survival Guide

Overview

Relational Database Software Powers Enterprise Applications

ERP

- General Ledger, Cash Management, Accounts Payable, Accounts Receivable, Fixed Assets, Human Resources, Payroll

CRM

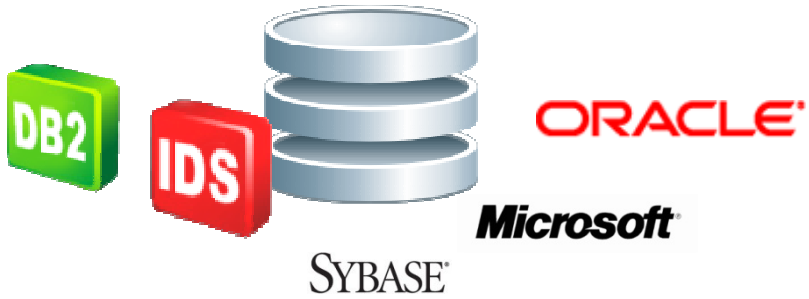
- Sales and Marketing, Commissions
- Service
- Customer Contact and Call Center support

Data Warehousing

- Canned reports
- Ad-hoc Reporting
- OLAP
- Data Mining

Leading Relational Databases Efficiently Support

- 100s to 1,000s of users
- Milliseconds to seconds response times
- 1,000s of transactions per minute



As Number of Users Increase and Data Volumes Grow
Data Management Performance Must Increase 10x

Communications

Financial Services

Web 2.0

Online Charging
- Authenticate and authorize
- Initiate service
- Manage credit balance
- Manage volume discounts

Brokerage Application
- Receive market feed
- Evaluate equity positions
- Check for fraud

Online Retail Web Site
- Authenticate user
- Manage personal wishlists
- Generate page contents with cross-sell data

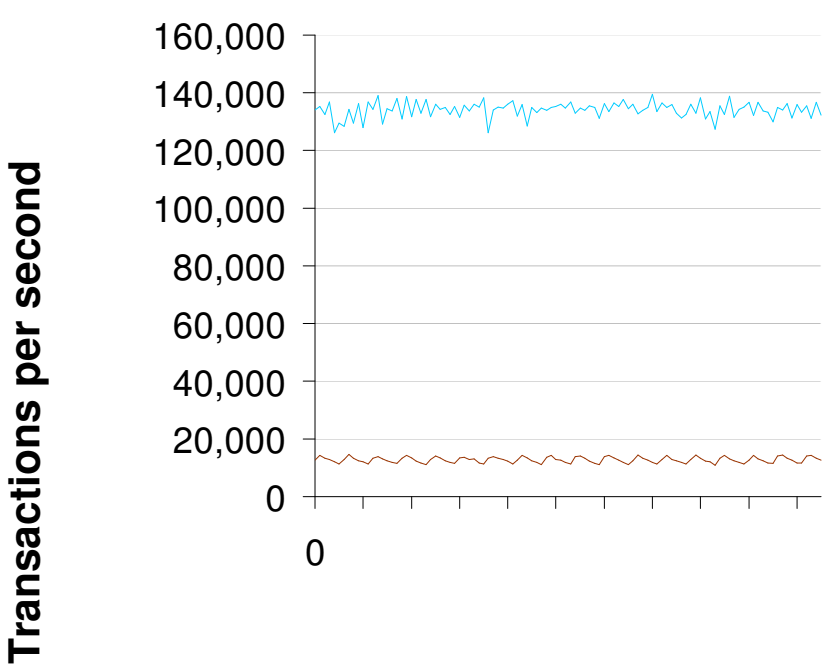
- 100,000s to **1,000,000s** of concurrent requests
- **10s of microseconds** for database calls

- Evaluate **30,000+ rules** on **500 trades per second** for 15 million trades per day

- Facebook: **10,000,000** concurrent sessions = two billion page views a day
- Wikipedia: 3000 page views a second and **25,000** SQL requests per second

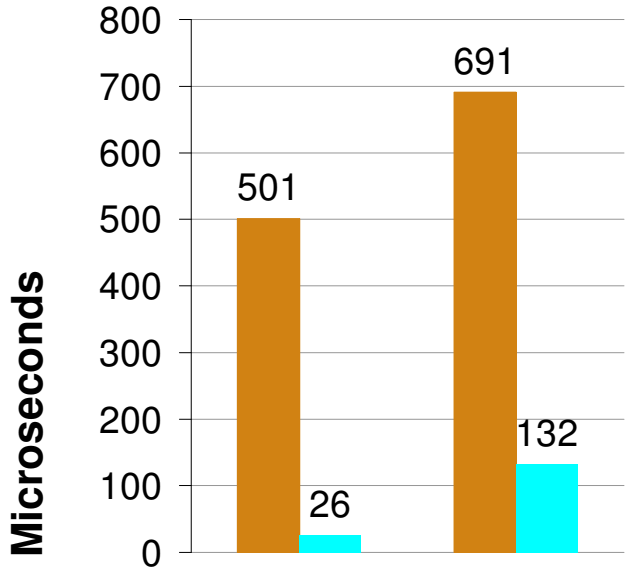
The Solution: Relational, In-Memory, Database Technologies Process Performance-Critical Data 10 times faster

Throughput of Tens of Thousands of Transactions per Second



- In-memory database
- Disk-based database

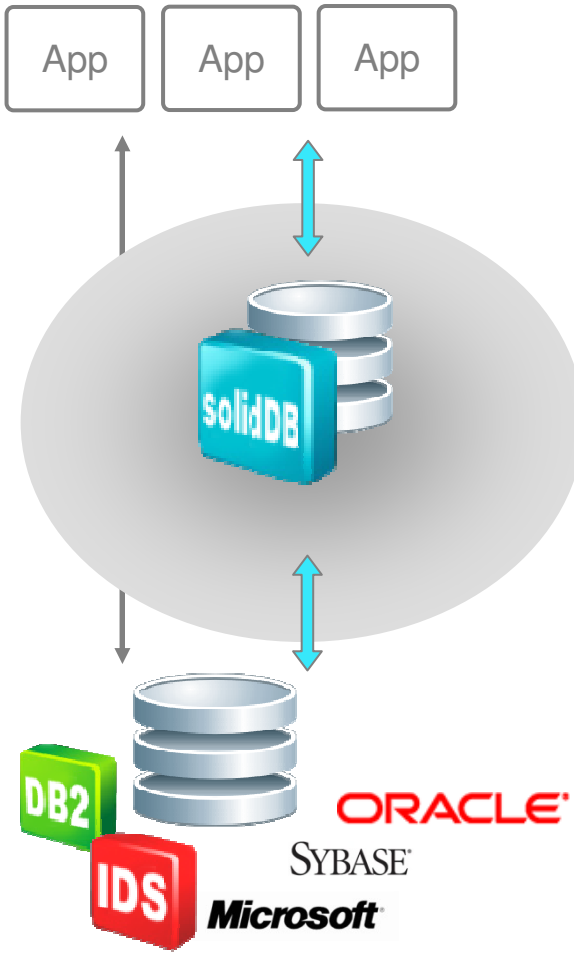
Response Times Measured in Microseconds



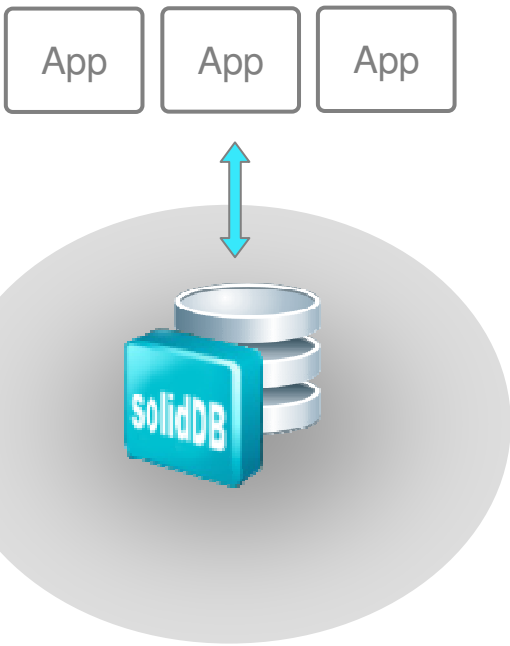
- In-memory cache + disk-based database
- Disk-based database

IBM solidDB 6.3 Product Family
Relational, In-Memory Database Technologies for Extreme Speed

IBM solidDB Universal Cache

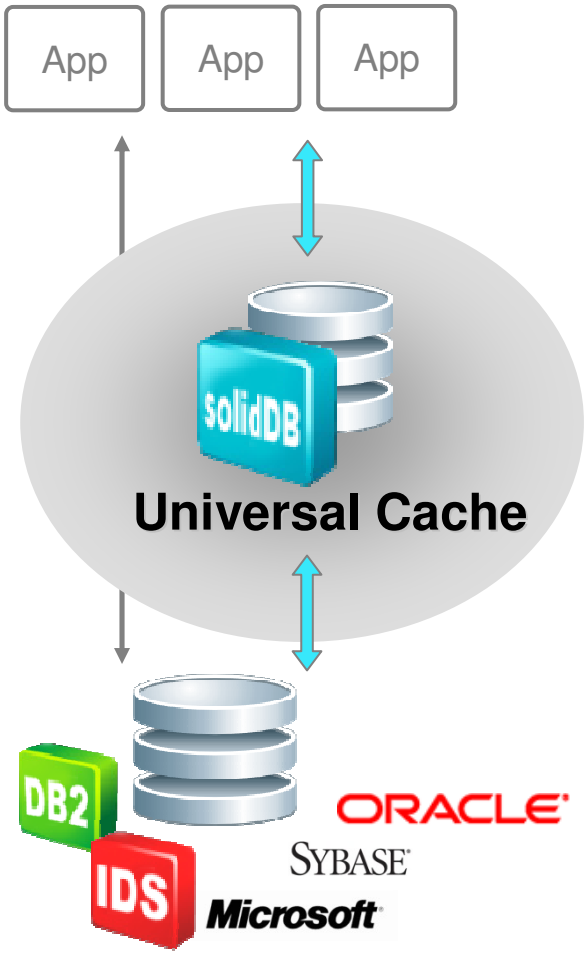


IBM solidDB



Announcing IBM solidDB Universal Cache

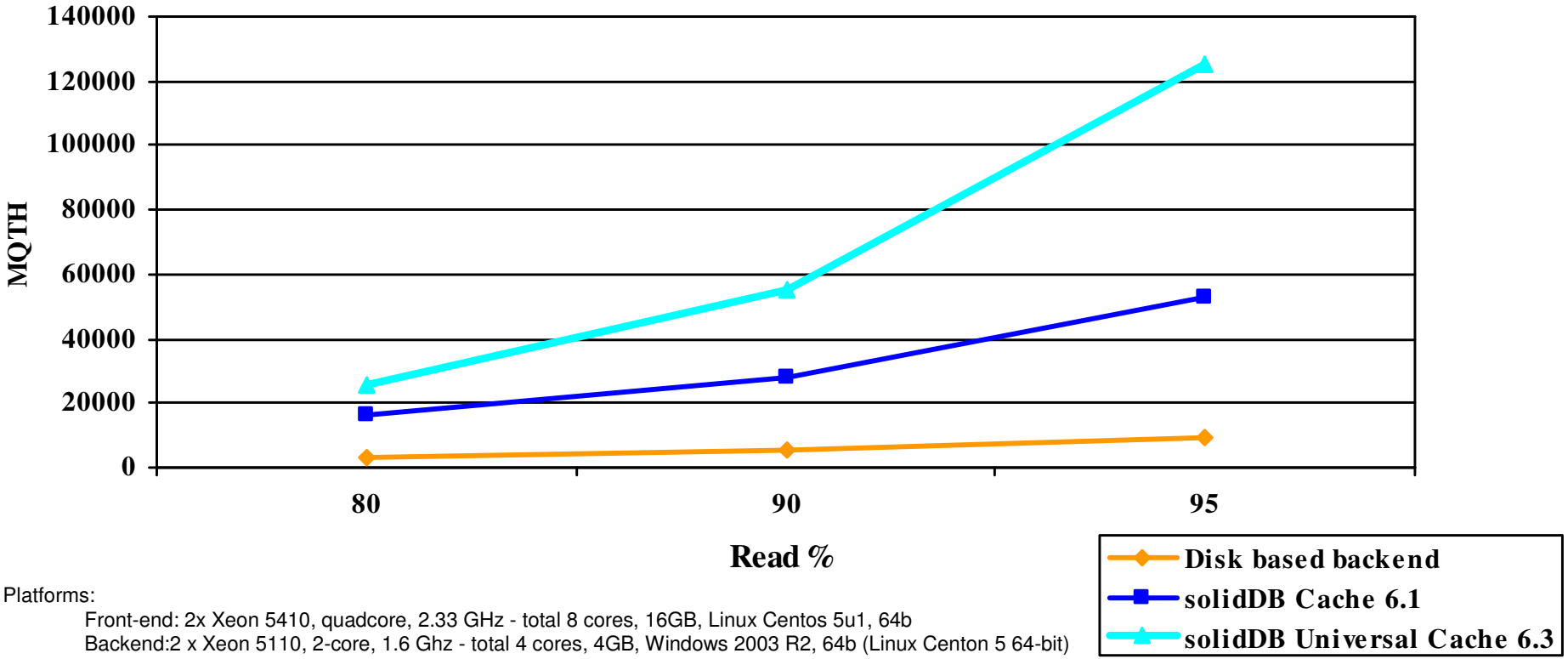
The industry's first relational, in-memory caching technology that accelerates IBM, Microsoft, Oracle and Sybase databases up to 10 times



- **Universal Caching**
  - Accelerates IBM DB2 for z/OS, DB2 for i/OS, DB2 for LUW, and IDS, Microsoft, Oracle, and Sybase
- **Extreme Speed**
  - Tens of thousands of transactions per second
  - Response times measured in microseconds
  - Brings performance-critical data closer to the application
- **Adaptability**
  - Adapts to different application and deployment needs
  - Provides powerful schema mapping and data transformation
  - Scales vertically and horizontally
- **Robustness**
  - Preserves data durability
  - Delivers high availability and instant failover

Preliminary solidDB Universal Cache 6.3 Performance Data

solidDB Universal Cache 6.3
Transaction Throughput



Platforms:
- Front-end: 2x Xeon 5410, quadcore, 2.33 GHz - total 8 cores, 16GB, Linux Centos 5u1, 64b
- Backend:2 x Xeon 5110, 2-core, 1.6 Ghz - total 4 cores, 4GB, Windows 2003 R2, 64b (Linux Centon 5 64-bit)
- Load: Remote with back-end and consolidated (local clients) with front-end

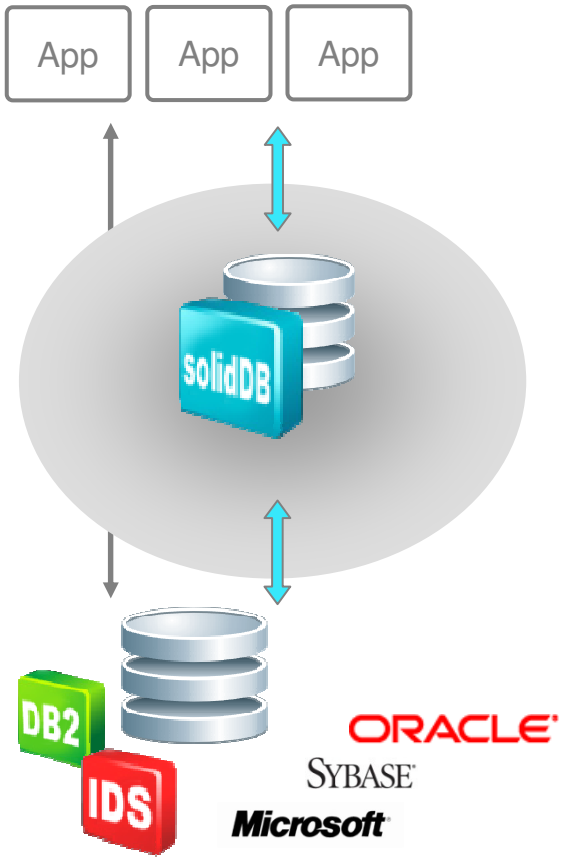
solidDB settings:
- Durability: relaxed, default checkpointing
- Isolation: Read Commited

TM1 settings:
- 1M Subscribers, 8 clients (100K subscribers, 8 clients)

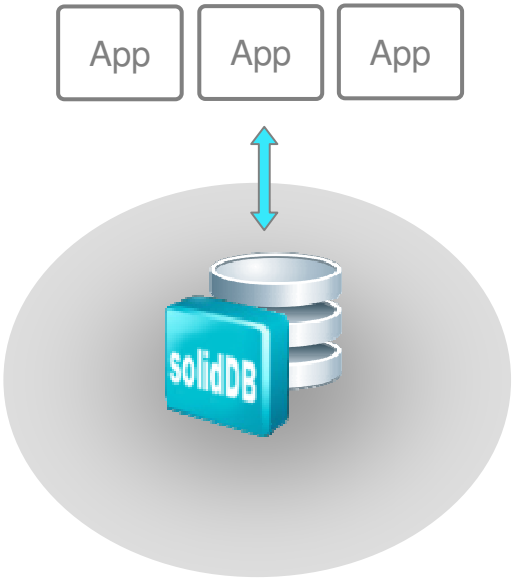
Packaging

IBM solidDB 6.3

IBM solidDB Universal Cache 6.3



IBM solidDB 6.3



IBM solidDB 6.3 Platform Support
solidDB and solidDB Universal Cache

OS	OS Details	Hardware
AIX	AIX 5L v5.3, v6	64-bit systems with POWER5 or POWER6 processors
Linux	RedHat Enterprise Linux (RHEL) 4 and 5 SUSE Linux Enterprise Server (SLES) 9 and 10	32-bit, 64-bit systems based on Intel or AMD processors
Solaris	Solaris 10	32-bit,64-bit systems with UltraSPARC or x86 processors
HP-UX	HP-UX 11i v2 and 11iv3	Itanium based HP Integrity Series systems
Windows	32-bit, 64-bit MS Windows Server 200 and 2003	32-bit,64-bit systems based on Intel or AMD processors

Back-end Database Support

Database	Version
DB2 for LUW	V9.1, V9.5
IDS	V11.50.3
DB2 for i/OS	V5, V6
DB2 for z/OS	V7,V8,V9
Oracle	9i,10g,11g
Sybase	V12.5.4, V15
Microsoft SQL Server	2000, 2005, 2008

Back-end Platform Support

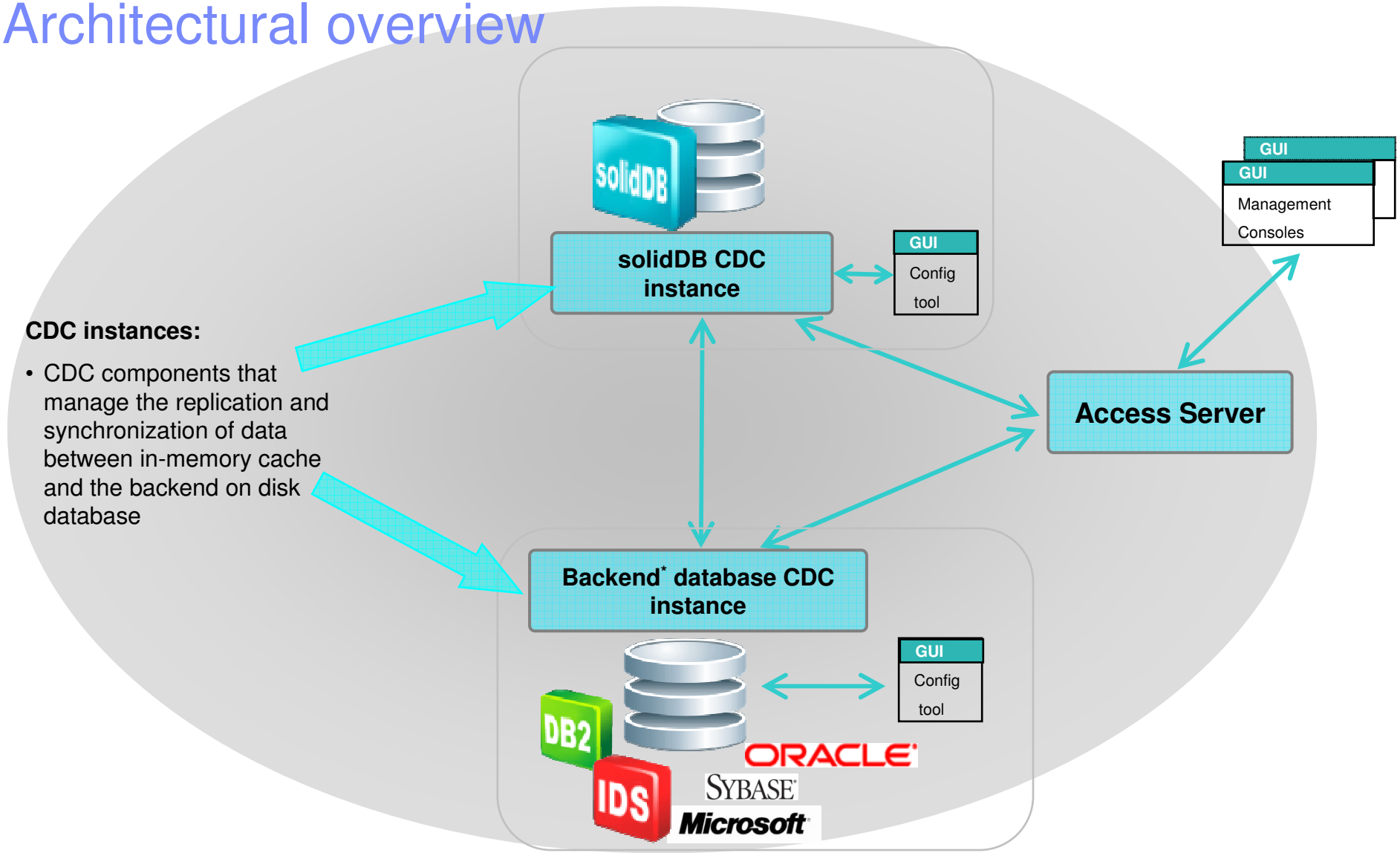
OS	OS Details	Hardware
z/OS	z/OS V1.4 or later	System z
AIX	AIX 5L v5.3, v6.1	64-bit systems with POWER5 or POWER6 processors
Linux	RedHat Enterprise Linux (RHEL) 4 and 5 SUSE Linux Enterprise Server (SLES) 9 and 10	32-bit, 64-bit systems based on Intel or AMD processors
zLinux	RedHat Enterprise Linux (RHEL) 4 and 5 SUSE Linux Enterprise Server (SLES) 9 and 10	64-bit System z
Solaris	Solaris 10	32-bit,64-bit systems with UltraSPARC processors
HP-UX	HP-UX 11i v2 and 11iv3	Itanium based HP Integrity Series systems
Windows	32-bit, 64-bit MS Windows Server 200 and 2003	32-bit,64-bit systems based on Intel or AMD processors

What's New in Universal Cache 6.3?

solidDB Universal Cache 6.3 Architectural overview

CDC instances:

- CDC components that manage the replication and synchronization of data between in-memory cache and the backend on disk database



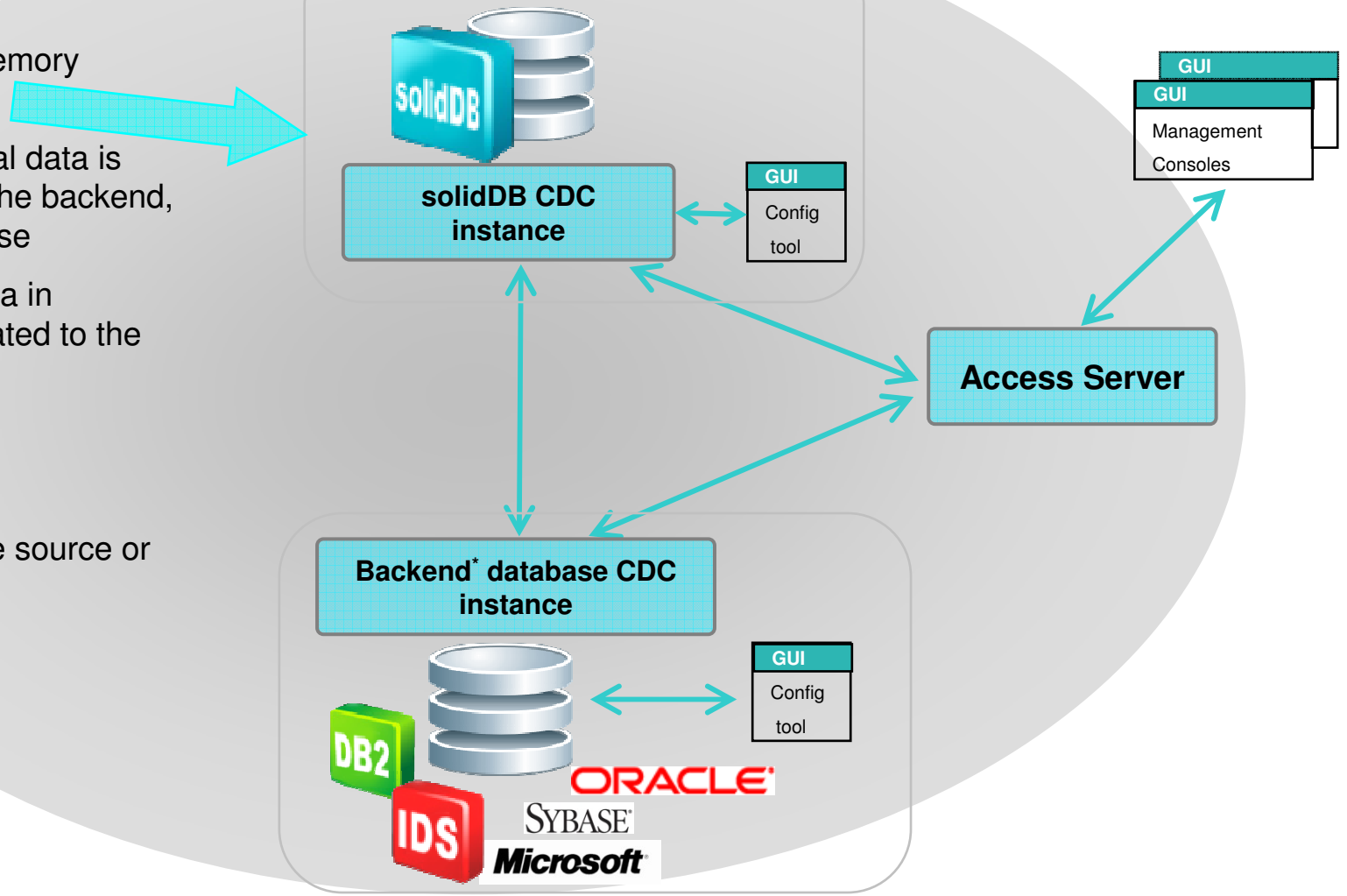
solidDB Universal Cache 6.3
Architectural overview

solidDB:

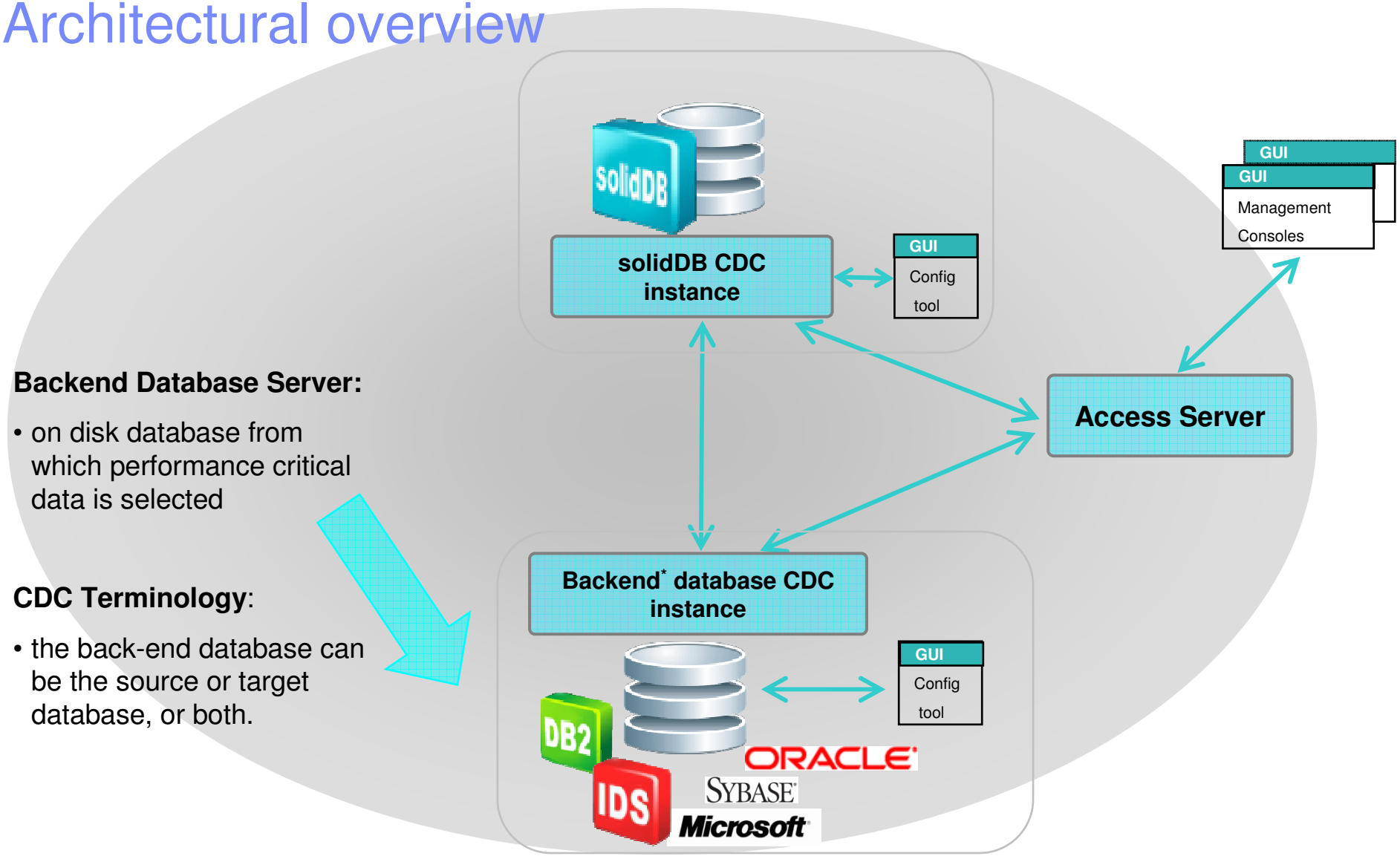
- the front-end in-memory cache
- performance critical data is loaded here from the backend, disk based database
- changes to the data in solidDB are replicated to the backend

CDC Terminology:

- solidDB can be the source or target or both



solidDB Universal Cache 6.3
Architectural overview



Backend Database Server:

- on disk database from which performance critical data is selected

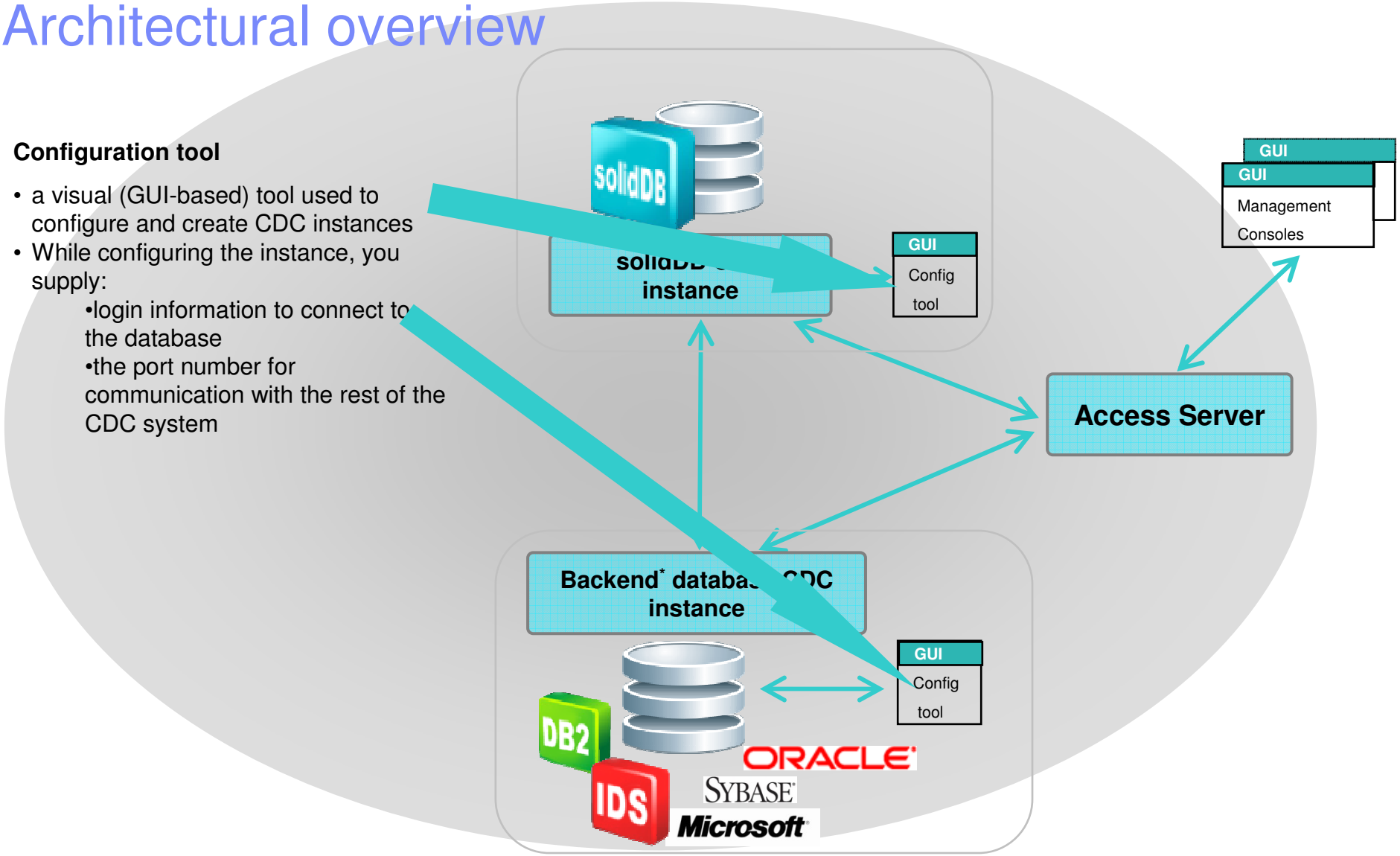
CDC Terminology:

- the back-end database can be the source or target database, or both.

solidDB Universal Cache 6.3
Architectural overview

Configuration tool

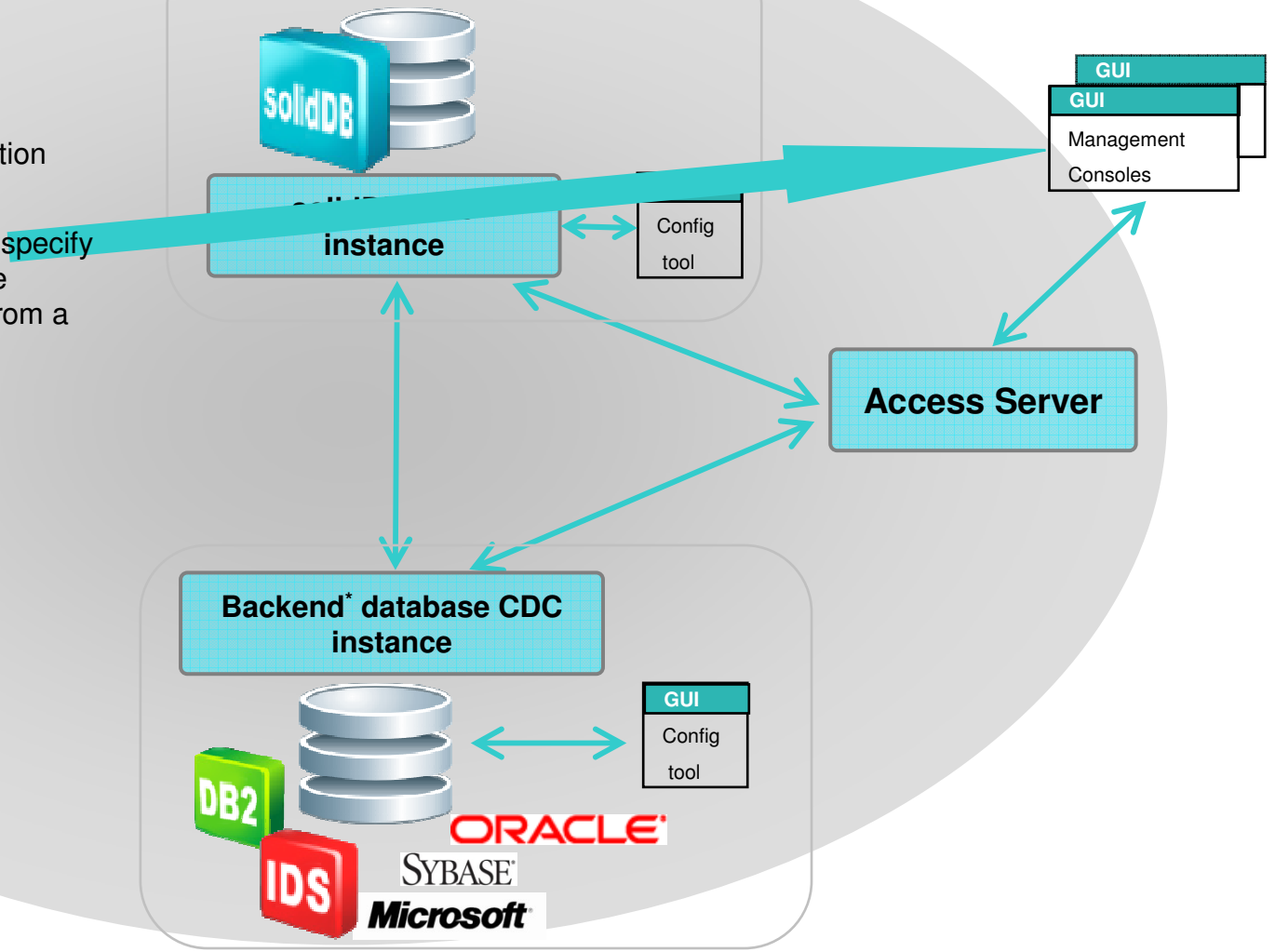
- a visual (GUI-based) tool used to configure and create CDC instances
- While configuring the instance, you supply:
  - •login information to connect to the database
  - •the port number for communication with the rest of the CDC system



solidDB Universal Cache 6.3
Architectural overview

Management Console

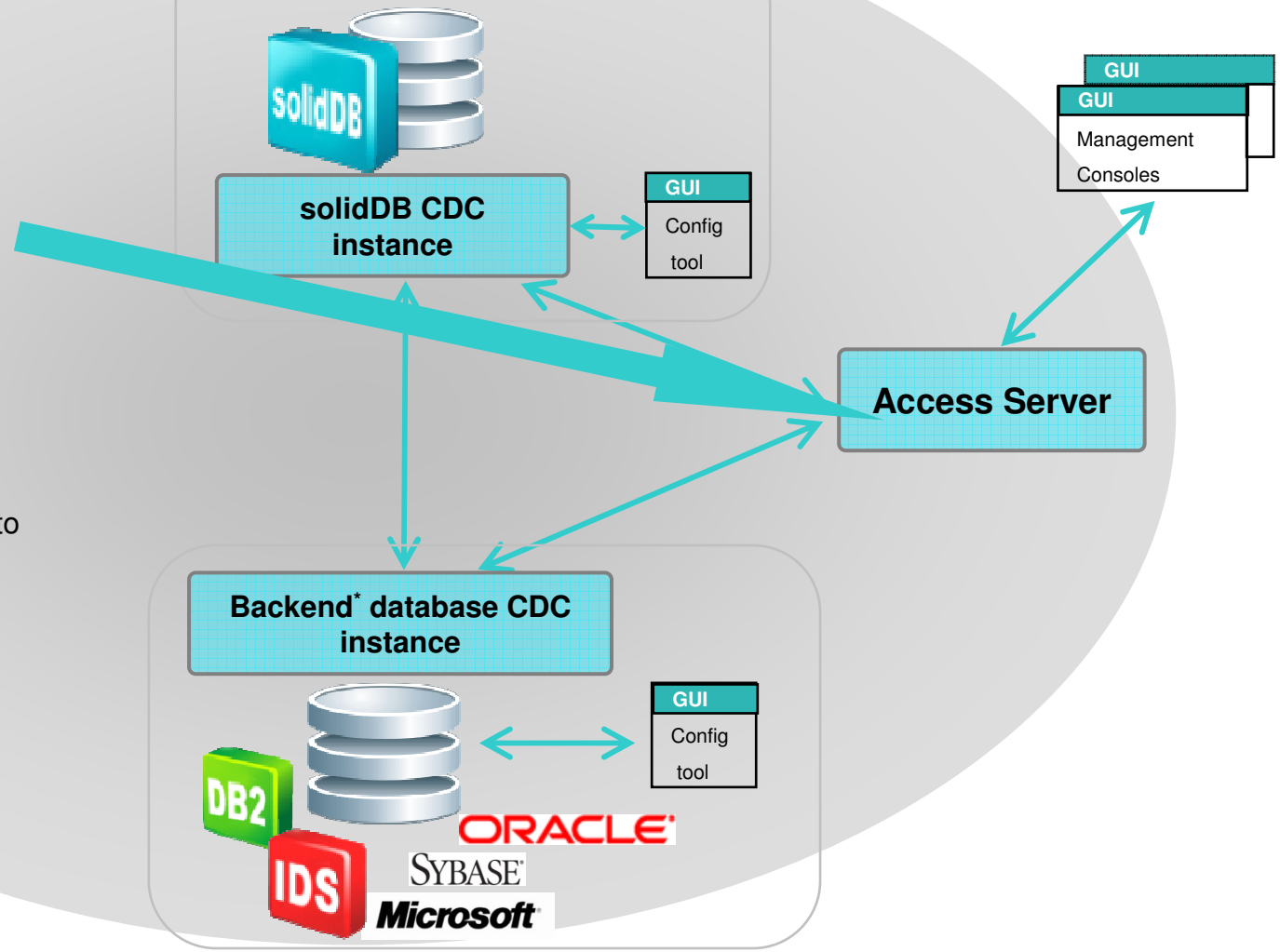
- an interactive GUI application for configuring and monitoring replication and synchronization.
- allows you to manage replication, specify replication parameters, and initiate refresh and mirroring operations from a client workstation



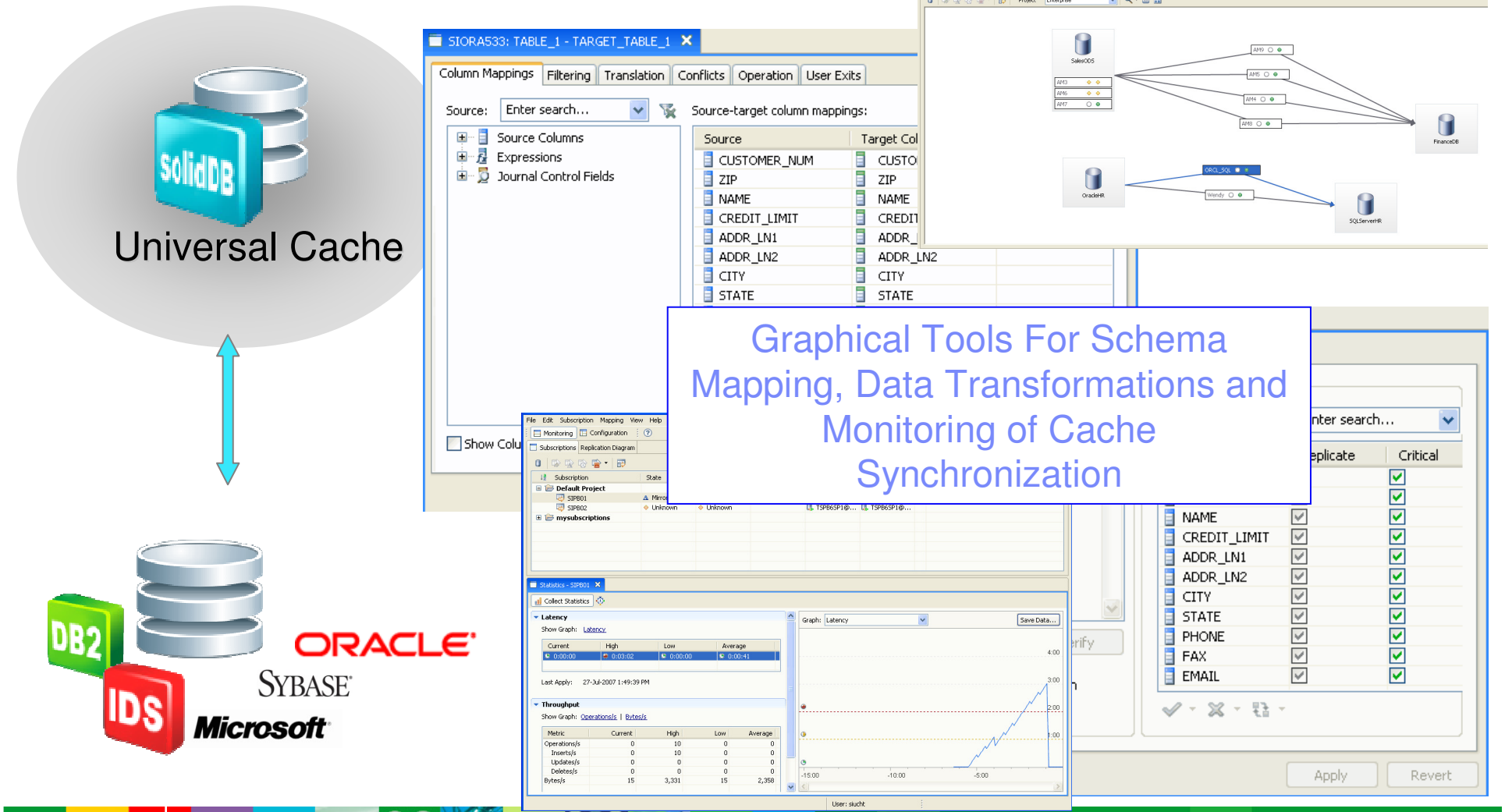
solidDB Universal Cache 6.3
Architectural overview

Access Server:

- a process (running as service or daemon) allowing the Management Console users to access the CDC instances and configure them. Different users may have access to different instances.
- During the Access Server installation you are requested to provide a port number and Administrator login information, to be used by the Management Console

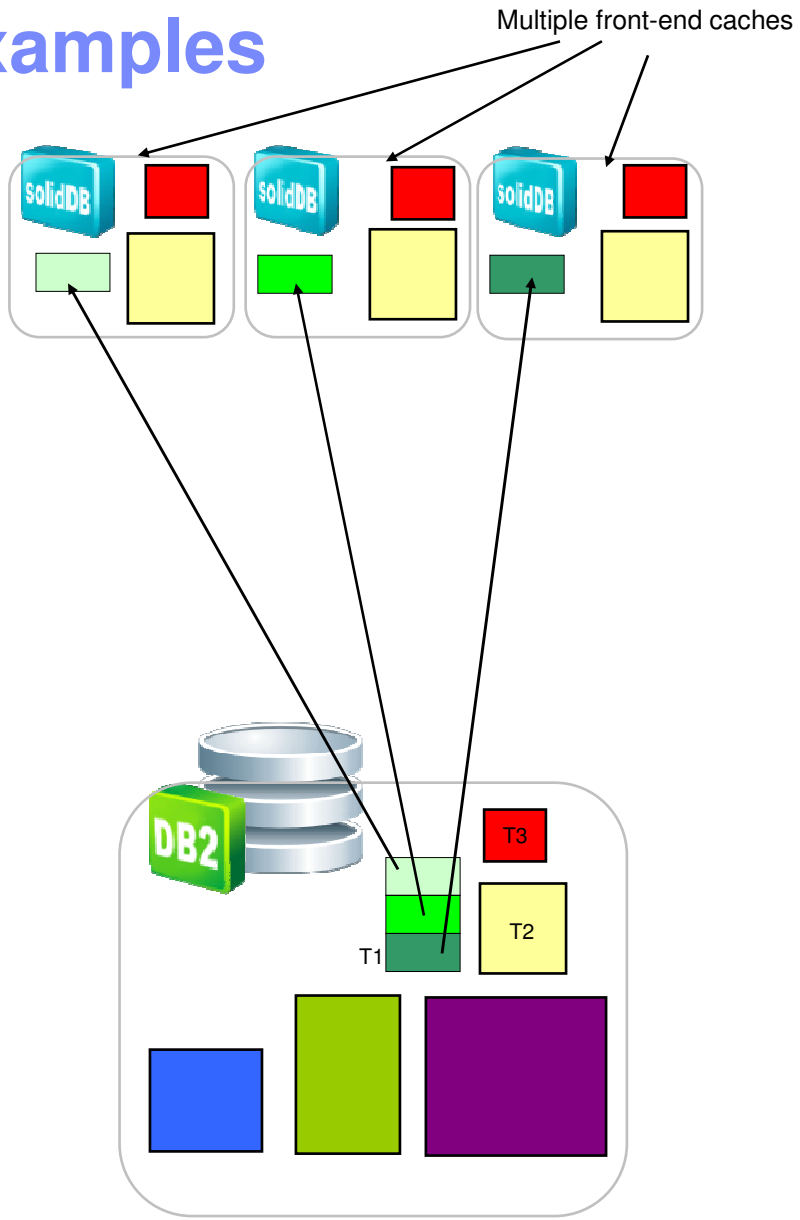


Adaptability: What tools are available?



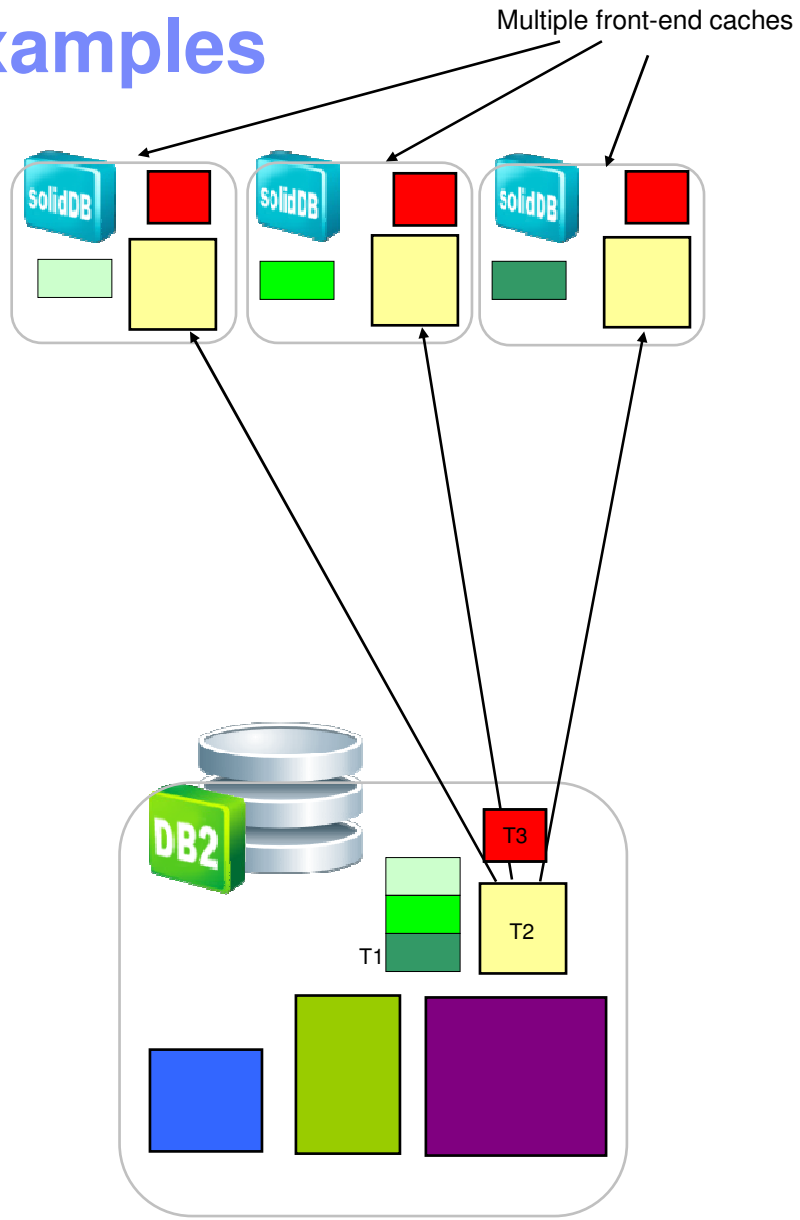
Typical In-Memory Caching Examples

- A backend table **T1** is horizontally partitioned (by way of a key), and each partition is loaded into solidDB in-memory caches
- The partitions are primarily updated in the in-memory cache
  - updates in-memory are replicated to the backend automatically
  - there are no update conflicts if the data is not updated at the back-end.
  - an **update conflict resolution mechanism is available** should there be a need to allow for back-end originated updates



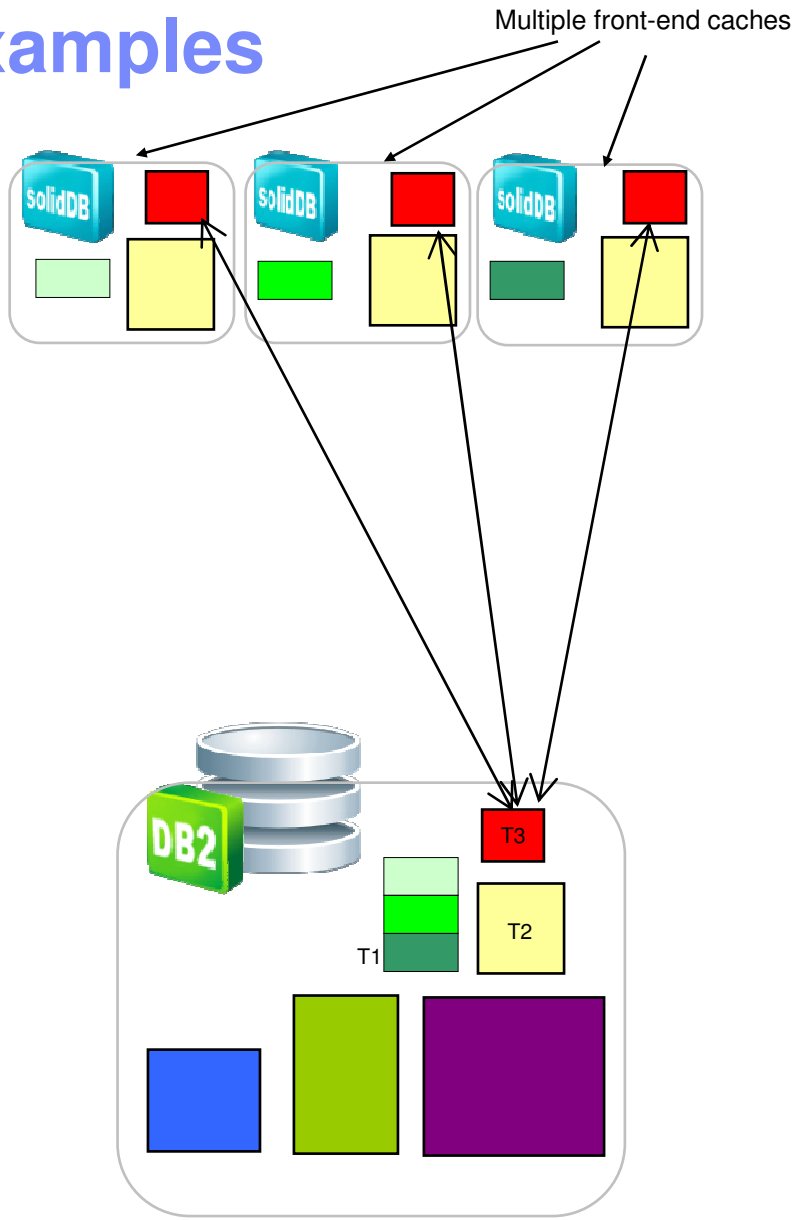
Typical In-Memory Caching Examples

- The backend table T2 is loaded into solidDB in-memory caches for read-only use
- T2 is only updated at the backend
  - Changes in the backend are replicated to the in-memory cache



Typical In-Memory Caching Examples

- Table T3 is loaded into all solidDB in-memory caches
- If updates are allowed at all nodes, as well as the backend database, there is a conflict resolution mechanism available.



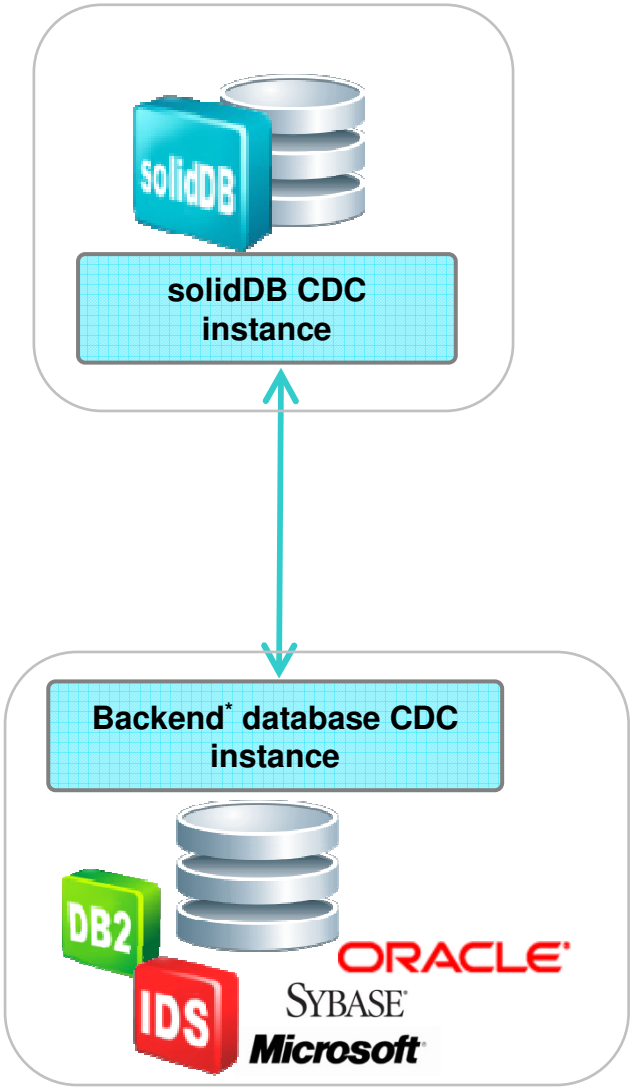
Handling Failures in non-HA Configurations

If any of the components fails

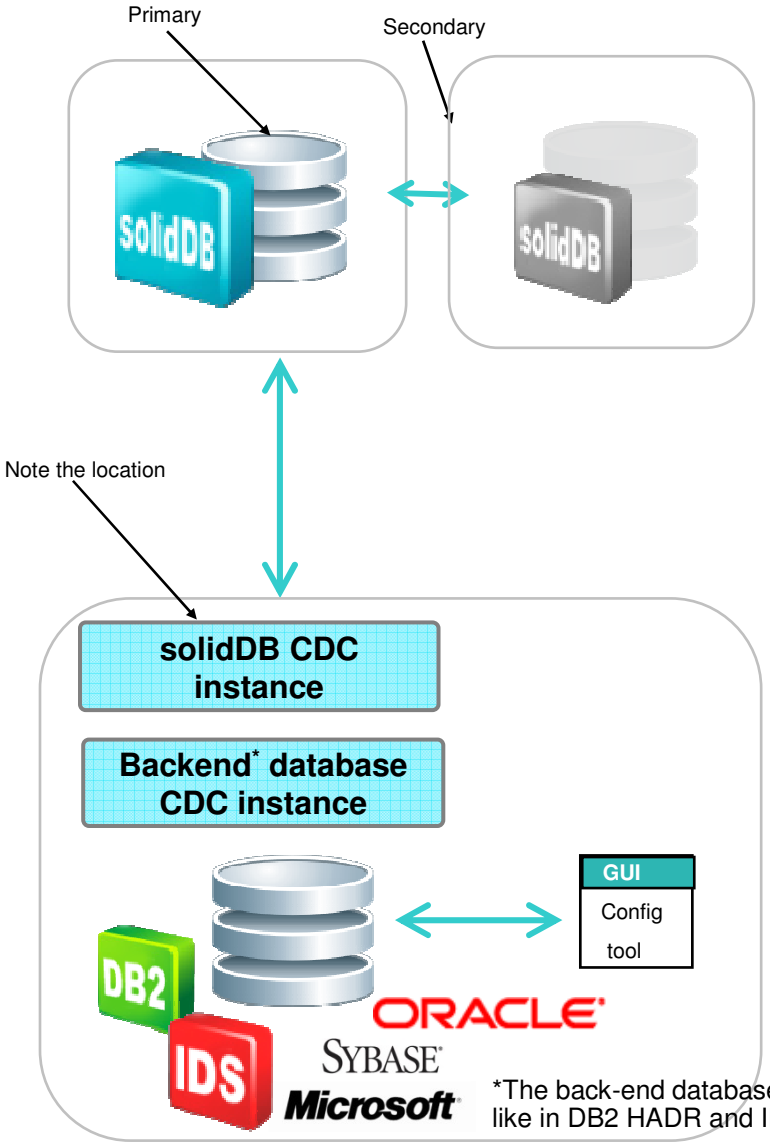
1. Restart the failed component
2. Stop and restart both CDC instances
3. Activate subscription mirroring

In most cases databases are re-synchronized and replication continues normally

In some cases, a subscription refresh may be needed



Universal Cache & High Availability



Features:

- Subsecond failover
- Transparent Connectivity
- Load balancing

Deployment:

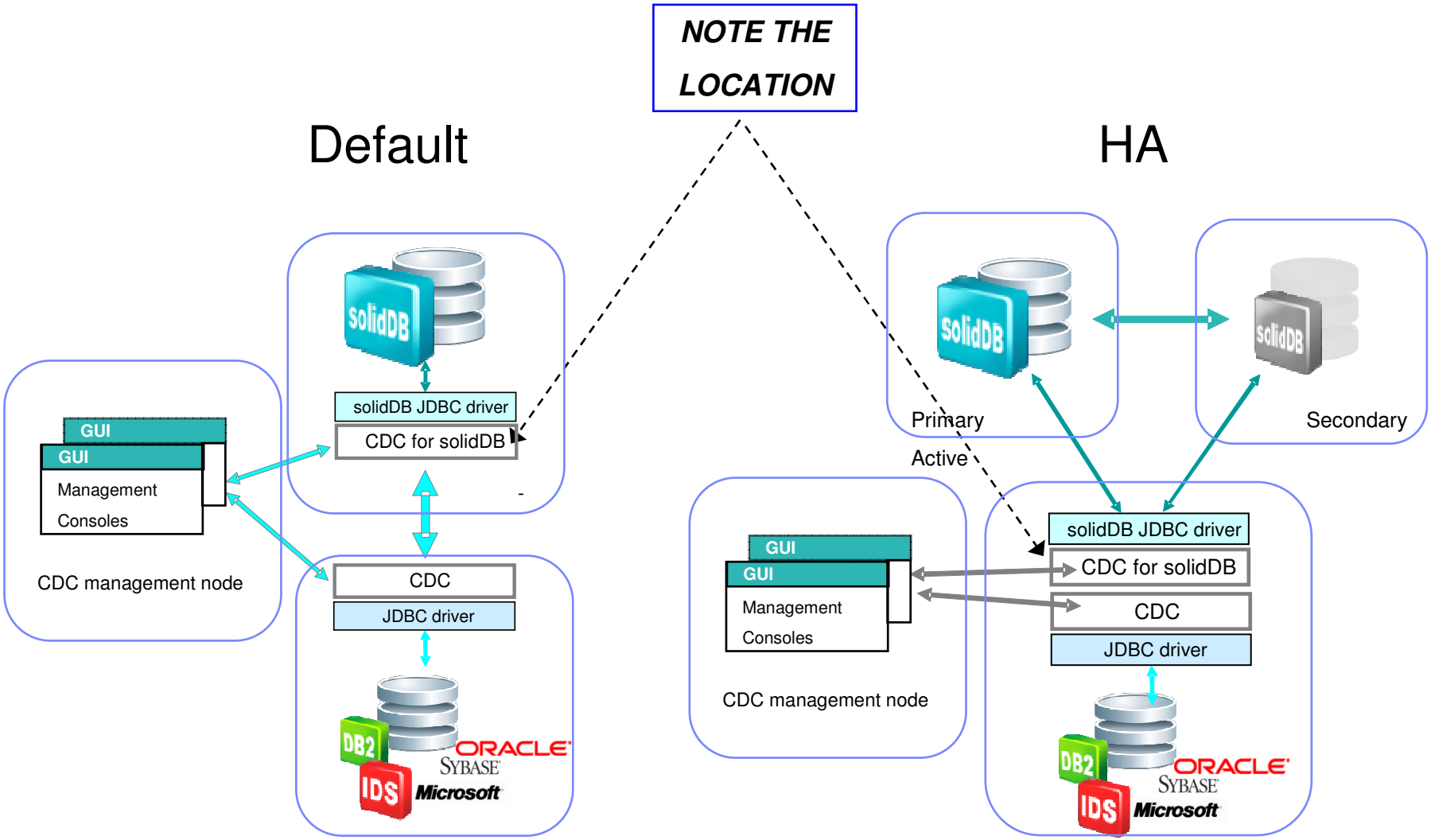
- CDC for solidDB instance is deployed on the back-end so that CDC system is capable to recover from failures in the front-end tier

Failovers:

- via HA Controller or external HA Manager
- CDC components for solidDB are HA-aware
- A CDC instance may fail during failure case scenarios. It is assumed that, when restarted, the CDC instance is capable of recovering fully the replication state and continue without loss of data.

*The back-end database deployment may also involve an active-standby configuration, like in DB2 HADR and IDS HDR. A N-active configuration is possible with Oracle RAC.

Universal Cache Deployment with High Availability



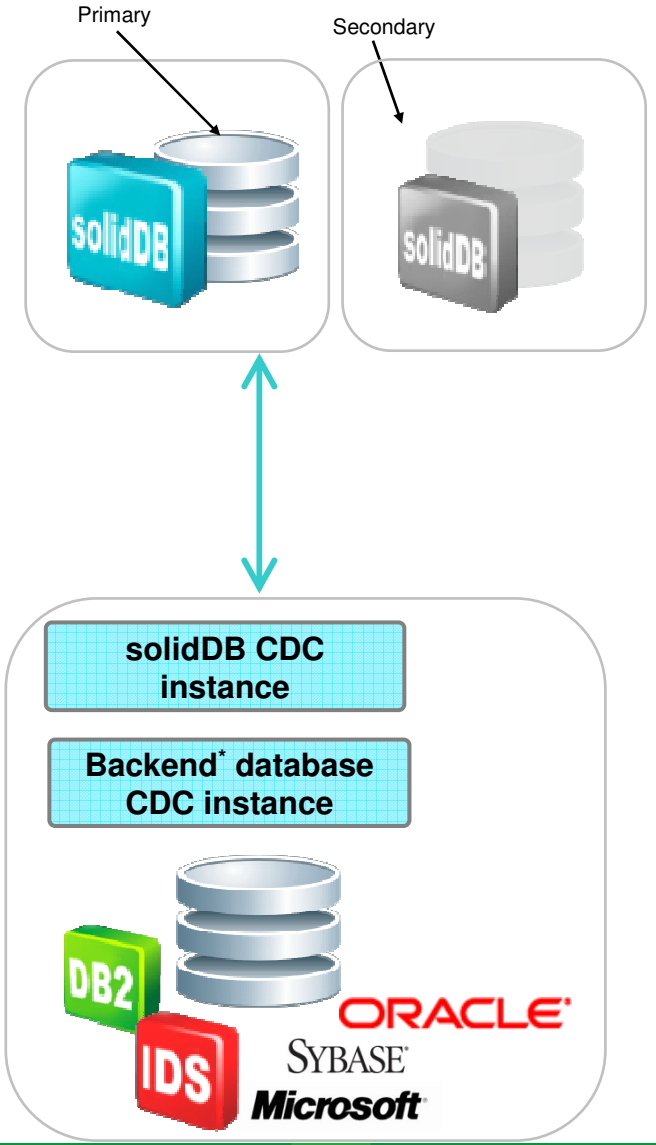
Handling failures when solidDB HA (HotStandby) is used

Primary solidDB front-end fails

- IBM solidDB High-Availability Controller (HAC), in the front-end tier performs a **fail-over to the secondary** front-end as a standard procedure.
- If the 2-Safe protocol is used, the database and log states are fully preserved.
- The applications perceive a failover time of less than one second.
- If solidDB is configured as a source, the CDC components failover automatically to the new Primary.
- If solidDB acts as a target, the mirroring stops at the failover. It is then restarted with a CDC command (or an automated script) executed at the source side. Mirroring continues normally.

Secondary solidDB front-end fails

- **No intervention is needed** to continue with the replication.
- Secondary front-end node is recovered in a normal way (for example, automatically rebooted). HAC automatically performs the rest of the recovery.
- The failure is not visible to applications or to the CDC for solidDB.



Handling failures when backend HA is used

- In the case of a failure in the backend:
  - the database recovery is governed by the backend management policies and tools
  - solidDB server does not offer any aids to assist in the back-end system recovery.
- Once the backend server is back up and running
  - an exact copy of the CDC instance is restarted at the new/restarted server
  - CDC for solidDB will need to be reconfigured to connect to the new peer
    - You may automate the process of reconfiguring by using CDC Java APIs
- In some cases, a subscription refresh may be needed.

What's New in solidDB 6.3?

IBM solidDB 6.3: What's New

- Performance improvements
- UTF-8 support
- New configuration parameter defaults
- New solidDB-to-solidDB replication technology
  - Enables fast data replication leveraging solidDB's transaction log API
  - Leverages same data mapping and data transformations capabilities, and graphical tools as in solidDB Universal Cache
  - Allows multiple standby solidDB instances to replicate data from one solidDB hot-standby pair in one data center to another one
  - Enables two solidDB instances to be configured in an active/active setup similar to TimesTen's, to enable high availability, as an alternative to solidDB's hot-standby configuration