SM1.4/AS4.20 Research and Development in Nuclear Explosion Monitoring

Poster B299
EGU2015-3323

IDCDACS: IDC's Distributed Application Control System

Martin ERTL[1], Alexander BORESCH[1], Ján KIANIČKA[1], Alexander SUDAKOV[2], and Elena TOMUTA[2]

(1) Angewandte Wissenschaft Software und Technologie (AWST) GmbH, Mariahilfer Str. 47/3/1, A-1060 Vienna, Austria

(2) Preparatory Commission for the Comprehensive Nuclear-Test-Ban Treaty (CTBT) Organization, Vienna International Centre, P.O. Box 1200, A-1400 Vienna, Austria

Time-series Data from S/H/I Sensors

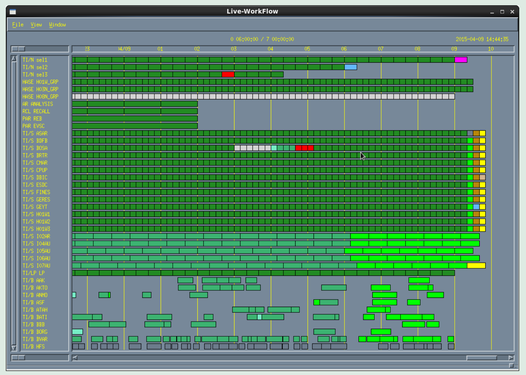

Figure 1: Time-series data from global network of seismic, hydro-acoustic and infrasound (SHI) sensors processed at the International Data Centre (IDC). Showing processing status per Time Interval (TI).

Layered Software Architecture

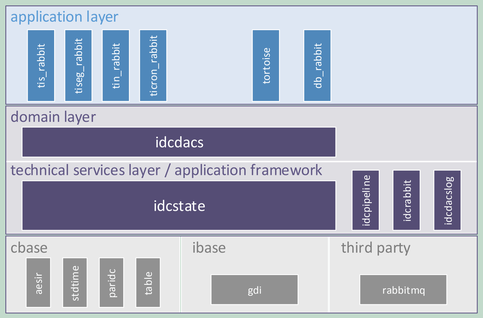

Figure 2: Application Control Server Programs (blue) are built on top of a reusable software framework (purple) encompassing domain and technical services layers. All applications and libraries are implemented in the C programming language.

Software Framework / Object-Oriented Design

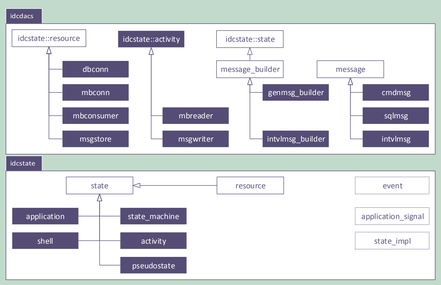

Figure 3: Framework libraries provide building blocks in form of interfaces, classes and configurable objects. They implement common functionality like application initialization, main control loop, exceptions, state machine, resource management, transactional messaging, and client-side failover for high-availability cluster. Bottom up, object-oriented design leads to simple, analyzable and maintainable code.

Setting the Scene

The Preparatory Commission for the CTBTO is an international organization based in Vienna, Austria. Its mission is to establish a global verification regime to monitor compliance with the Comprehensive Nuclear-Test-Ban Treaty (CTBT), which bans all nuclear explosions. For this purpose time series data from a global network of seismic, hydro-acoustic and infrasound (SHI) sensors are transmitted to the International Data Centre (IDC) in Vienna in near-real-time, where it is processed to locate events that may be nuclear explosions.

See Figure 1.

Accomplishments

We newly designed the distributed application control system that glues together the various components of the automatic waveform data processing system at the IDC (IDCDACS). Our highly-scalable solution preserves the existing architecture of the IDC processing system that proved successful over many years of operational use, but replaces proprietary components with open-source solutions and custom developed software. Existing code was refactored and extended to obtain a reusable software framework that is flexibly adaptable to different types of processing workflows.

See Figure 2 and Figure 3.

Conceptual Design

Automatic data processing is organized in series of self-contained processing steps, each series being referred to as a processing pipeline. Pipelines process data by time intervals, i.e. the time-series data received from monitoring stations is organized in segments based on the time when the data was recorded. So-called data monitor applications queue the data for processing in each pipeline based on specific conditions, e.g. data availability, elapsed time or completion states of preceding processing pipelines.

See Figure 4.

System Design and Implementation

IDCDACS consists of a configurable number of distributed monitoring and controlling processes, a message broker and a relational database. All processes communicate through message queues hosted on the message broker. Persistent state information is stored in the database. A configurable processing controller instantiates and monitors all data processing applications. Due to decoupling by message queues the system is highly versatile and failure tolerant.

The implementation utilizes the RabbitMQ open-source messaging platform that is based upon the Advanced Message Queuing Protocol (AMQP), an on-the-wire protocol (like HTML) and open industry standard. IDCDACS uses high availability capabilities provided by RabbitMQ and is equipped with failure recovery features to survive network and server outages. It is implemented in C and Python and is operated in a Linux environment at the IDC.

See Figure 5, Figure 6 and Figure 7.

Configurability, Versatility and Adaptability

Although IDCDACS was specifically designed for the existing IDC processing system, its architecture is generic and reusable for different automatic processing workflows, e.g. similar to those described in (Olivieri et al. 2012, Kværna et al. 2012). Major advantages are its independence of the specific data processing applications used and the possibility to reconfigure IDCDACS for different types of processing, data and trigger logic. A possible future development would be to use the IDCDACS framework for different scientific domains, e.g. for processing of Earth observation satellite data extending the one-dimensional time-series intervals to spatio-temporal data cubes.

Processing Pipelines / Message Flow

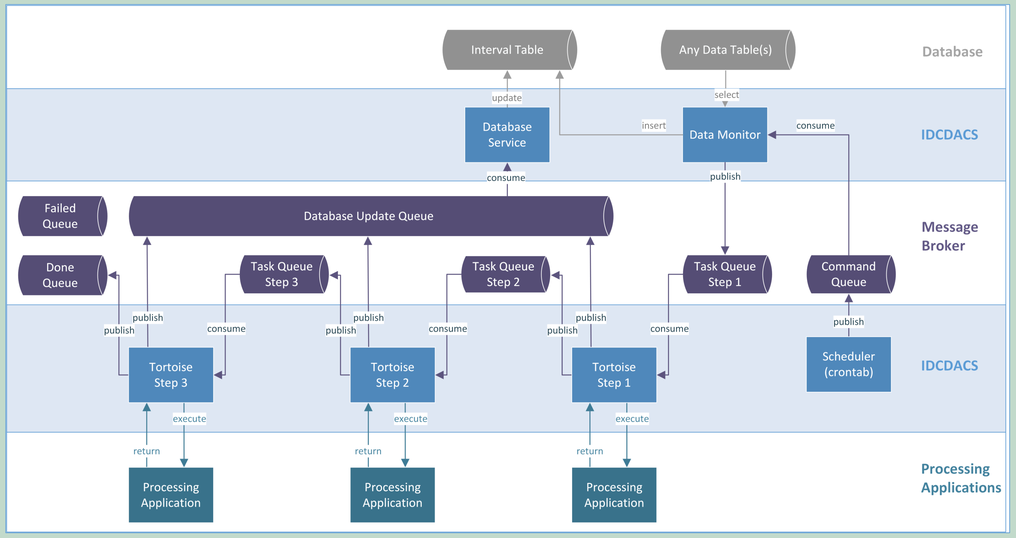

Figure 4: Example of a Processing Pipeline with three Processing Steps. Application Control Server Processes (blue) run as daemons listening on Message Queues (purple). The processing starts with Scheduler periodically publishing to Command Queue, waking up Data Monitor. Data Monitor checks conditions (data availability, status information, time, etc.) and creates Time Intervals by inserting into Interval Table and publishing to Task Queue Step 1. An instance of the Generalized Processing Server – Tortoise – executes the appropriate Processing Application (aqua), reports its return status to Interval Table through Database Update Queue, and publishes to next Task Queue. In this way Time Interval passes through all Steps of the Pipeline until reaching Done Queue. In case of a processing error Time Interval will be short-circuited into Failed Queue.

Deployment / Distribution

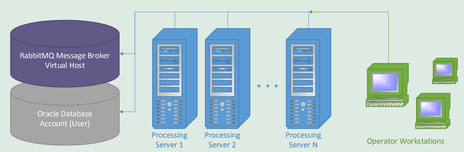

Figure 5: Nodes of IDCDACS: 1 Database, 1 Message Broker, 1-N Processing Servers hosting distributed Application Control Server Programs and Processing Applications. Operators connect through SSH and HTTP(S) from Operator Workstations.

Messaging / Decoupling / Parallelism

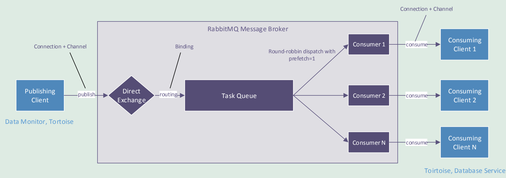

Figure 6: Task Queue decouples Publishing and Consuming Clients, leading to increased reliability and robustness, and permits parallel processing – for load balancing and scaling up.

Software Components and Processes

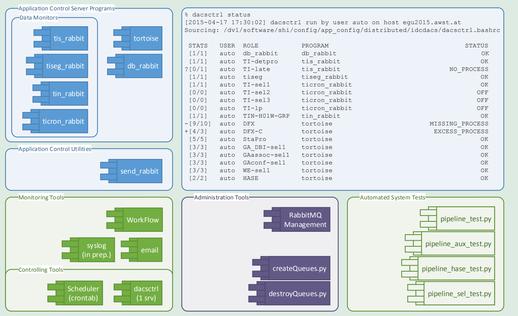

Figure 7: Application Control Server Programs are monitored and controlled with dacsctrl/Scheduler, Message Queues with RabbitMQ Management Web Console, Time Intervals with WorkFlow (cf. Figure 1). The System is validated with Automated System Tests.

Development Status

A first major Release is running on the IDC development LAN. Transition to Testbed and Operations LANs is in progress. A successor project is planned to improve usability, to extend features, and to lift current limitations.

References

IDC Documentation: Distributed Application Control System (DACS) Software User Manual IDC-6.5.2Rev0.1, February 2000; and DACS Software Design Description, IDC-7.3.1, June 2001.

Olivieri M., J. Clinton (2012): An almost fair comparison between Earthworm and SeisComp3, Seismological Research Letters, 83(4), 720-727.

Kværna, T., S. J. Gibbons, D. B. Harris, D. A. Dodge (2012): Adapting pipeline architectures to track developing aftershock sequences and recurrent explosions, Proceedings of the 2012 Monitoring Research Review: Ground-Based Nuclear Explosion Monitoring Technologies, 776-785.

IDC Documentation: IDC's Distributed Application Control System (IDCDACS) Software User Manual; and IDCDACS Software Design Description, in preparation.

Angewandte Wissenschaft Software und Technologie (AWST) GmbH, Mariahilfer Str. 47/3/1, A-1060 Vienna, Austria
Telephone: +43-1-5861314 – Facsimile: +43-1-5861314-22 – Email Address: ertl@awst.at