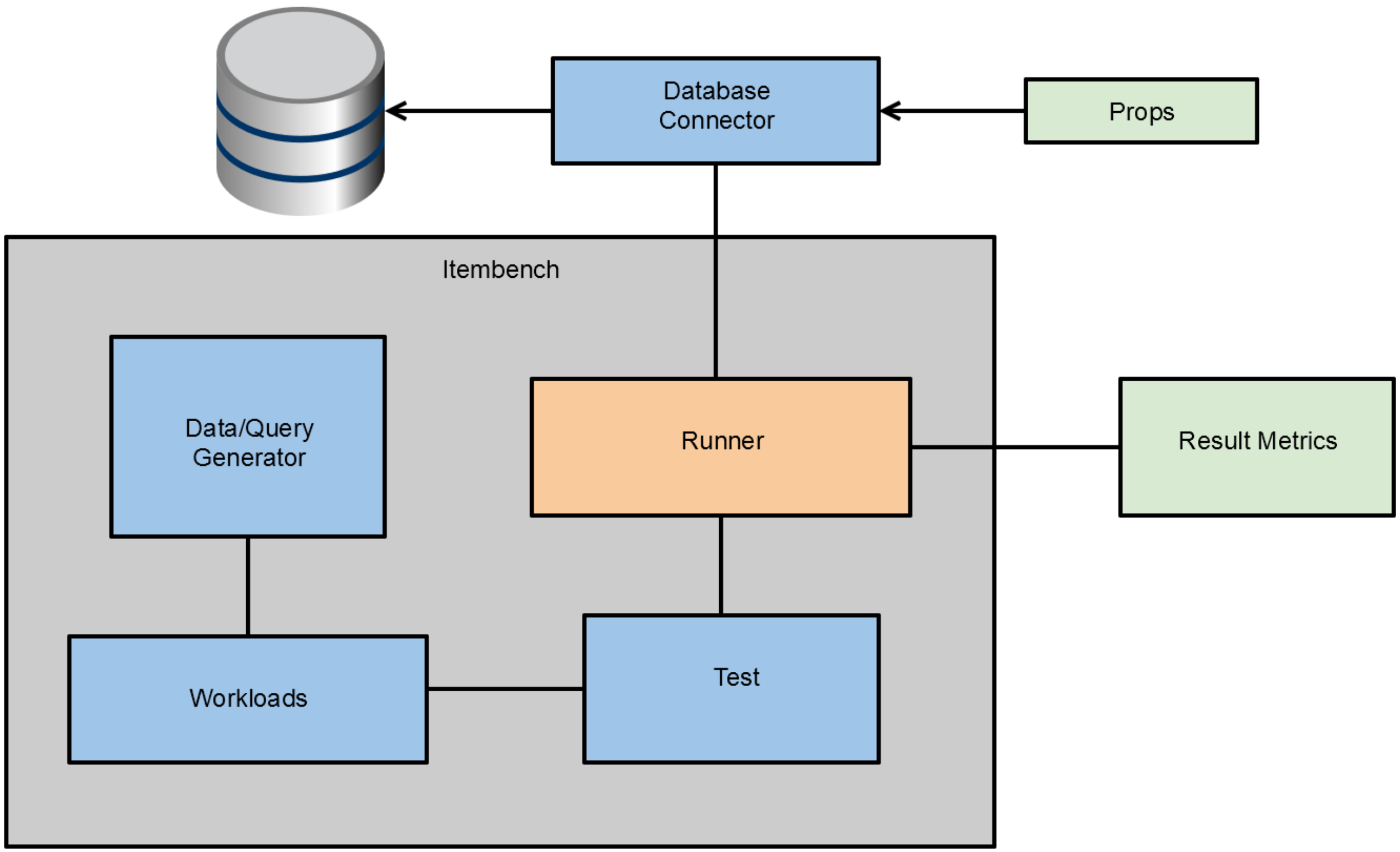
Embedded Database Benchmark

Team CodeBlooded

Architecture



Case Study - IoT Thermostat

- Model 1 – Application state storage
- Device used only to monitor temperature and humidity with time stamp
- Decision making onus shifted onto central RDBMS
- Workload type 100% insert

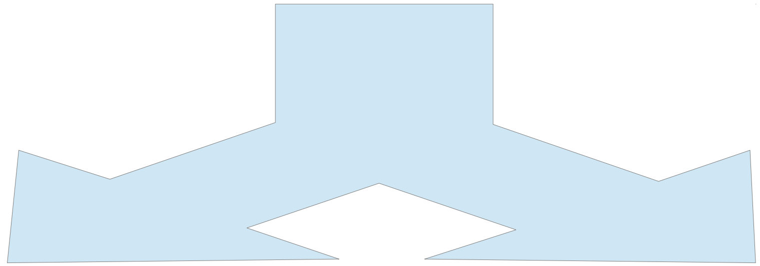
Case Study - IoT Thermostat

- Model 2 – Application state storage with programmatic decisions
- Device used to monitor temperature and humidity with time stamp
- Application keeps running values of all important variables and averages. On finding an outlier application performs an action. Eg Sending an email/dropping the temperature.
- Two sub implementations seen
  - DB stores only running values at a fixed interval (first insert followed by updates)
  - DB stores both running value at an interval and each reading for logging purposes (write dominant, insert heavy with updates performed at defined intervals)
- Workload types
  - 1 insert + rest update
  - Average case : temperature logged every 5 seconds, updates logged every minute
  - **Updates** = 1/13*100= 8% appx **Inserts =** 12/13*100 = 92% appx

Case Study - IoT Thermostat

- Model 3 – Modelling actions as SQL queries
- Application queries the database in order to check for outliers (read queries)
- Actions are performed with the help of after insert triggers
- Values for outliers in the selection predicate are predefined by application programmer
- Workload type 50% insert, 50% read
  - Insert and select happen at the same time due to the trigger monitoring the application
  - Updates may or may not happen depending on implementation. The number however is marginal and the overall workload is dominated by the above two queries
- Type of select query – Considering a thermostat that works records temperature and humidity the select query for threshold monitoring is a simple select with an attached where clause
- Possible complex query – select query which checks if the average temperature and average humidity over the entire day is above predefined thresholds

Case Study – Crowd Sensing



- **Mote Class Sensor Networks**

Mobile Crowd Sensing Networks

Traditional Mote Class Sensor Networks	Mobile Crowd Sensing
• Lesser number of sensors • Less resources such as compute power, memory. • Conditions or environment is comparably static.	• Large number of mobile sensors. No need to install, as they already exist in numbers. • More compute power, memory and communication resources. • Dynamic conditions – type of sensors, data quality energy level of device.

Local Analytics in Mobile Crowd Sensing

- Preprocessing of Raw Data to detect features.
  Eg: Pothole detection from 3-axis acceleration sensor data.
- Data Mediation: Filtering outliers, noise removal
- Context inference
  Eg: Kinetics mode of humans.

Resource Limitations

- Alternating between high quality and low quality sensor depending on energy levels.
- Variation of sampling rates according to priority.

Privacy Security

- Cryptography
- Adding random noise to mask user personal details

Model 1

Feature Detection & Context Inference

- Insert and Select do not happen at same time
- Insert 70 %
- Select 30%
- Complexity of query : Complex

Model 2

Data Mediation

- Insert and Select do not happen at same time
- Insert 50 %
- Select 50%
- Complexity : Simple to complex. Mostly threshold based.

References:

Mobile Crowdsensing: Current State and Future Challenges -
Raghu K. Ganti, Fan Ye, and Hui Lei