A novel system to integrate DSHIP data from marine research vessels

Maximilian Betz 3.9.25



© GEOMAR, Maike Nicolai (CC-BY 4.0)



© AWI / Ralf Röchert (CC-BY 4.0)



© AWI / Mario Hoppmann (CC-BY 4.0)



© Ein Dahmer (CC BY-SA 4.0)



© Uni Hamburg / LDF / Sören Janssen



© AWI / Folke Mehrtens



© Uni Hamburg / LDF / Tim Kalvelage



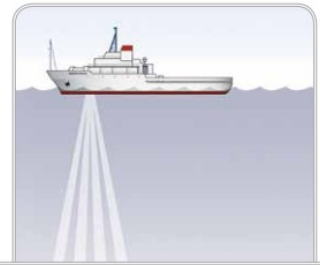
© Uni Hamburg / LDF / Stefan Seidel

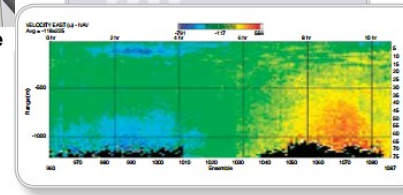
A quick wrap up

- De facto standard on all major German research vessels for managing discrete time series (streaming) data and Logging of scientific activities (ActionLog)
- Recorded Data
  - ActionLog Data which logs scientific activities, stations and gear operations. Like a stationbook
  - System and Vessel status: Pump Flow meters, Winch usage (tension, length),
  - Navigation GNSS, Motion Sensor, depth
  - Oceanographic: Salinity, Temperature, pCO2, Chlorophyll, O2, pH, Turbidity, Current velocity, ...
  - Weather: Pressure, Ceiling, Radiation, Dewpoint, Visability, Wind,
  - Environment Aerosol Spectrometer (Dust Monitor)
- Recorded data is, in a delayed mode, transferred to shore after an expedition and made available publicly.
- Continuous Data since: 1988 (Polarstern), 2005(Heincke), 2006(Maria S. Merian), 2008(Alkor), 2009(Meteor), 2012 (Sonne, Elisabeth Mann Borgese), 2013 (MYA), 2018 (Uthoern)



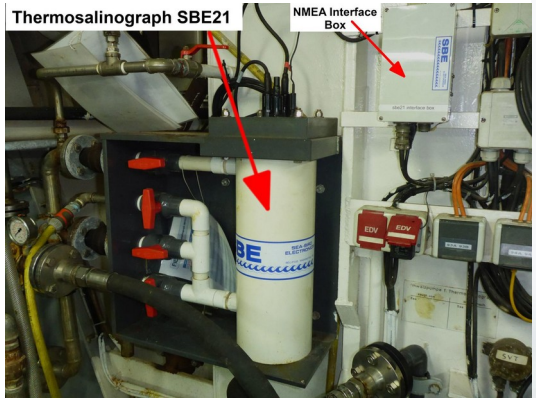
Ultrasonic anemometer Sonic 2D portside





SMP21 Pyranometer

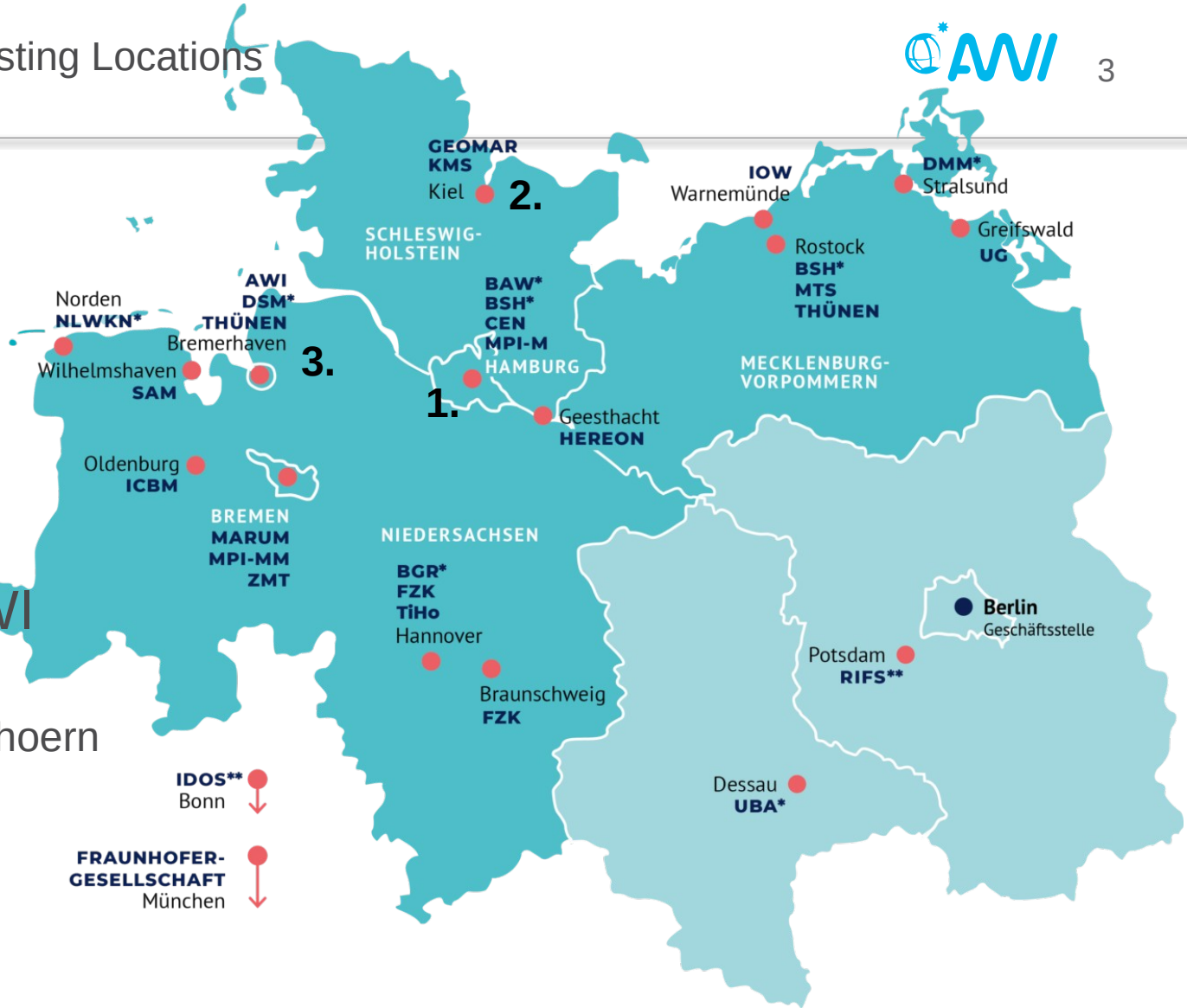






Landsystem - previous Data Hosting Locations

1. Hamburg BSH
   - Atair, EMB, Meteor, Merian, Sonne
2. Kiel Geomar
   - Atair, Poseidon
3. Bremerhaven AWI
   - Polarstern, Heincke, Neumayer, Mya II, Uthoern



- Data from distributed Landsystems and different versions (V2 (until 2016), V3 (after 2016) is consolidated and integrated into a single new system.
- Replaces legacy systems. All existing inventory data is harmonized and migrated.
- Legacy data access via Web Extraction is still supported
- Based on a scalable Kubernetes architecture
- Developed by WERUM & Hosted at AWI



- Data access across platforms possible through single access point.
- Apache data analytics stack
- Apache Parquet File based
- Apache Hive for metadata management
- API Data access via Trino
- Available via dship.awi.de planned: dship.o2a-data.de

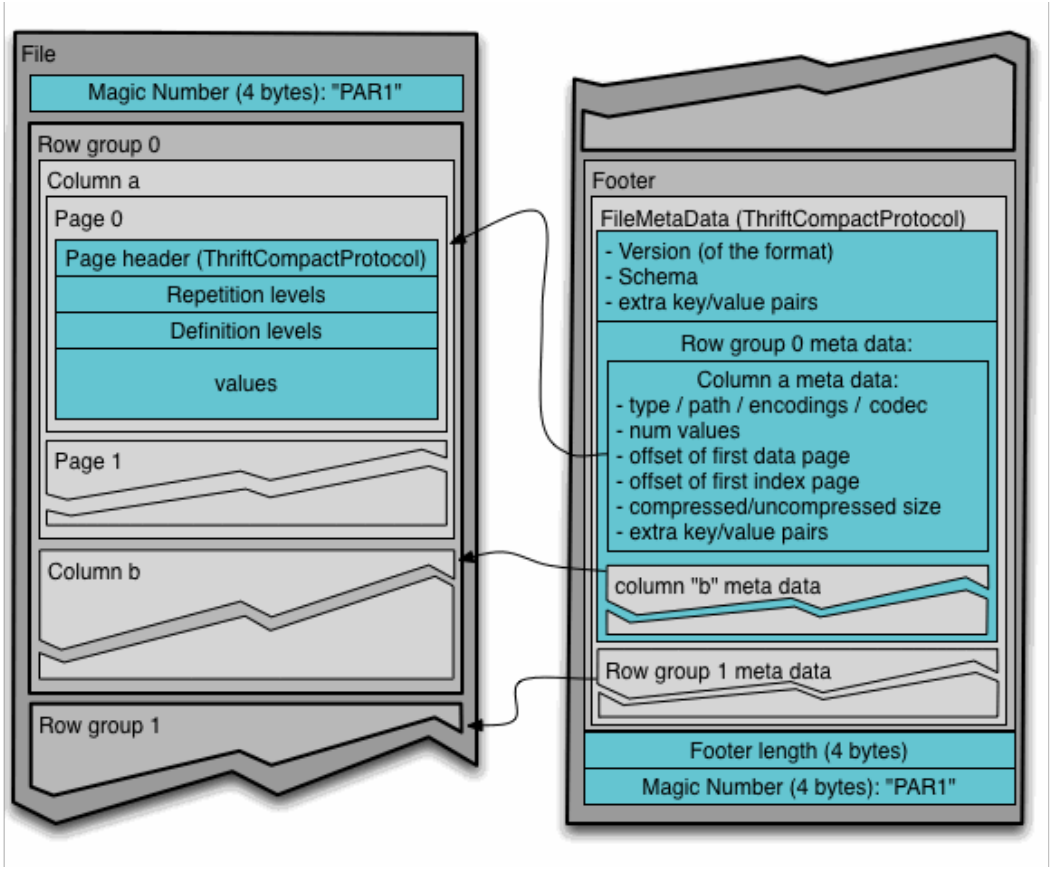






trino

- 2013 initiated by Twitter and Cloudera
- Binary format with column based storage and compression for efficient access of selected parameters
- Optimized for parallel and column based access via “Row-Groups”
- Supported by Python, C++, Rust, Java, …



A widely supported format. Low level data access is very simple

```
import pandas as pd
#WEATHER.PAWIBWW.water_temperature PS142 Parquet file
file = 'data/part-00000-04b43578-394a-4f5f-8565-d4145afa9ba2.c000.snappy.parquet'
data = pd.read_parquet(file)

print(data.columns)
print(data[['utc', 'latitude', 'longitude', 'value_double']].head(2))
```

```
mbetz@binf02l092:~/Documents/projects/data-logistics-support/scripts$ python3 parquet_example.py
Index(['dship_system', 'server_name', 'platform_name', 'data_source',
       'expedition_key', 'utc', 'data_type', 'value_double', 'value_text',
       'status', 'unit', 'physical_quantity', 'longitude', 'latitude'],
      dtype='object')
                        utc   latitude  longitude  value_double
0 2024-04-14 06:04:00.250803+00:00 -22.904903  14.498848          16.9
1 2024-04-14 06:05:00.248120+00:00 -22.907013  14.498680          16.9
```

- Trino provides a way to access the entire dataset via a single sql query
- A Hive Metastore provides a data index as metadata to trino about the relevant parquet files and data structure
- Trino scales by massively scheduling queries parallel to processing worker nodes
- Many options for query parameters. E.g. geopolynom, time-range, value limits, down sampling,
- Query examples
  - select * from dp.dship.tbl_sensor_data where (longitude is NULL or latitude is NULL) and (parameter_id like 'vessel:polarstern:dship1:system.speed') and utc >= timestamp '1999-01-01 00:00:00.000000' and utc <= timestamp '1999-12-31 00:00:00.000000'
  - WITH numbered_rows AS (select ROW_NUMBER() OVER (PARTITION BY parameter_id, date_trunc('HOUR',utc) ORDER BY utc) AS row_num, utc, latitude, longitude, parameter_id, data_source, unit, value_double from dp.dship.tbl_sensor_data where (parameter_id like 'vessel:polarstern:dship1:thermosalinometer_keel.temperature_keel' or parameter_id like 'vessel:polarstern:dship1:WEATHER.PAWIBWW.water_temperature') and utc >= timestamp '1999-01-01 00:00:00.000000' and utc <= timestamp '1999-12-31 00:00:00.000000') select * from numbered_rows where row_num = 1 order by utc



trino

- Creation of Trino SQL queries is complex
- A Python API query wrapper helps accessing the data pool in a simple way
- Installable via **pip install git+https://gitlab.awi.de/data-logistics-support/de.awi.datapool_trino_queries**
- There will be a demo session later

```python
from trino_query import pool
from datetime import datetime
import plotly.express as px

# Get the ActionLog in a pandas dataframe format
df_actionlog = pool.get_action_log('PS143/2')

# Get all available parameters
df_parameters = pool.get_sensor_parameter()

""" Get values from the following parameters down sampled to one value per hour in a pandas dataframe.
    Ignore 0.0 values or values outside of the range -10 ... 40
"""
parameters = [
'vessel:polarstern:dship1:thermosalinometer_keel.temperature_keel',
'vessel:polarstern:dship1:WEATHER.PAWIBWW.water_temperature'
]
df_sensor_values = pool.get_sensor_values_down_sampled('HOUR',
                                    parameters,
                                    min_value=-10, max_value=40,
                                    ignore_values=[0.0],   # ignore 0.0 as such values are likely incorrect
                                    start_time=datetime(1999,1,1),
                                    stop_time=datetime(1999,12,31)
                                    )

# Plot results in a map
df_sensor_values['info_data'] = df_sensor_values['utc'].astype(str) + '  ' + df_sensor_values['parameter_id'].astype(str)
fig = px.scatter_geo(df_sensor_values, lat='latitude', lon='longitude', color='value_double', hover_data='info_data')
fig.show()
```

- Integration of inventory data from all other major German research vessels
- Putting the data into Registry Devices / Parameter relations for enhanced metadata.
- A KI Chatbot for a further simplified access omitting complex Trino SQL queries.
- Live Data from vessels instead of delayed mode uploads which can take many weeks until data is available

- Feedback or Questions?
- Feel free to contact us via o2a-support@awi.de

References

- https://www.werum.de/de/insightics
- https://dship.awi.de/
- https://gitlab.awi.de/data-logistics-support/de.awi.datapool_trino_queries/
- https://trino.io/
- https://parquet.apache.org/
- https://www.databricks.com/glossary/what-is-parquet
- https://www.allianz-meeresforschung.de/app/uploads/dynamic/2024/09/240619-dam-mitgliederkarte-absep24-1344x0-c-default.png
- https://www.durag.com/en/product-filter-837.htm?productID=AirQualifier%20EDM%20264
- https://registry.o2a-data.de/rest/v2/items/625/images/154/payload
- https://spaces.awi.de/spaces/EFPW/pages/182354036/Weather+Station
- https://github.com/apache/parquet-format
- https://hive.apache.org/
- https://github.com/apache/parquet-format/blob/master/src/main/thrift/parquet.thrift