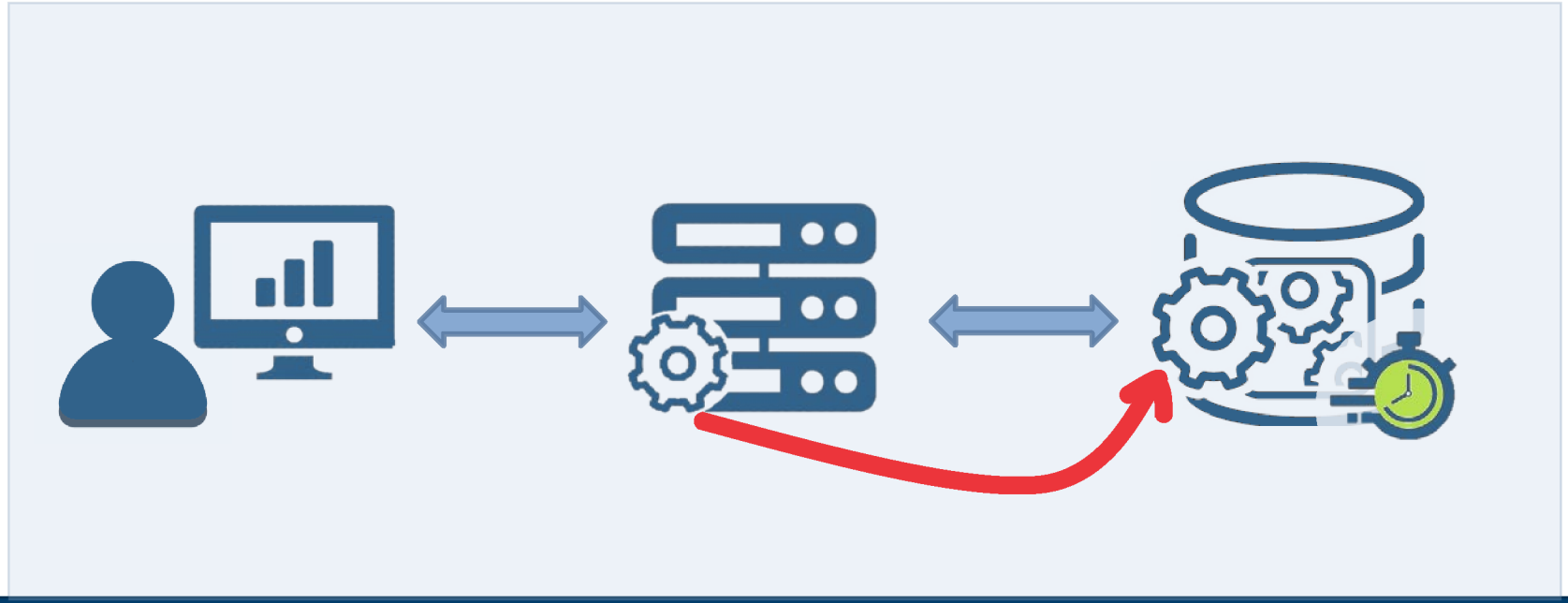


PERFORMANCE ANALYSIS AND OPTIMIZATION OF A TBYTE-SCALE ATMOSPHERIC OBSERVATION DATABASE

EGU 2020, ESSI2.11, 7 May 2020

Clara Betancourt, Björn Hagemeier, Sabine Schröder, Martin Schultz

Jülich Supercomputing Centre

Mitglied der Helmholtz-Gemeinschaft

IntelliAQ

TOAR tropospheric ozone assessment report

JÜLICH Forschungszentrum

“Hello, and thank you for considering this display.

Short summary: We present performance engineering of a TByte-scale air quality database (DB) system which was created by the Tropospheric Ozone Assessment Report (TOAR) and contains one of the world’s largest collections of near-surface air quality measurements. A special feature of our data service https://join.fz-juelich.de is on-demand processing of several air quality metrics directly from the TOAR database. As a service that is used by more than 150 users of the international air quality research community, our web service must be easily accessible and functionally flexible, while delivering good performance. The current on-demand calculations of air quality metrics outside the database are identified as the major performance bottleneck. In this study, we therefore explore and benchmark in-database approaches for the statistical processing, which result in performance enhancements of up to 32%. We will furthermore show how the web service infrastructure can be extended in functionality, allowing the calculation of flux-based ozone metrics.

The work is mainly based on our paper (Betancourt et al., in preparation). If you have questions, email me! **c.betancourt@fz-juelich.de**. I am looking forward to seeing you online, at our session!”

Performance analysis and optimization of a TByte-scale atmospheric observation database

Introduction: Fast access to scientific data products

- Performance of scientific database applications
- TOAR-DB service infrastructure

TOAR-DB performance benchmarks

- Overview
- Data aggregation inside versus outside the database
- Metrics calculation inside versus outside the database
- Parallel processing
- Influence of indices on query times
- Transfer times between database server and web server

Extend the functionality: Modeling global stomatal ozone fluxes

- Motivation: Ozone impact on vegetation
- Resources and realization

Summary and conclusions

Performance analysis and optimization of a TByte-scale atmospheric observation database

Introduction: Fast access to scientific data products

- Performance of scientific database applications
- TOAR-DB service infrastructure

TOAR-DB performance benchmarks

- Overview
- Data aggregation inside versus outside the database
- Metrics calculation inside versus outside the database
- Parallel processing
- Influence of indices on query times
- Transfer times between database server and web server

Extend the functionality: Modeling global stomatal ozone fluxes

- Motivation: Ozone impact on vegetation
- Resources and realization

Summary and conclusions

Performance of scientific database applications

Why is it important?

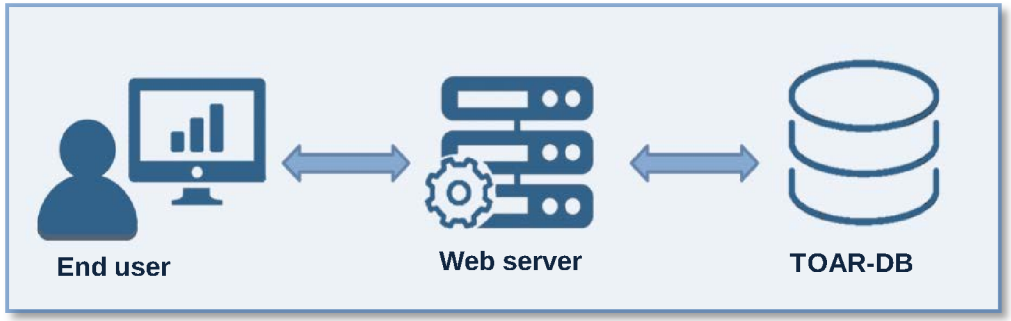
- Fast access to big data increases the quality, flexibility and outreach of scientific workflows, especially in interdisciplinary research
- Web interfaces to scientific databases enable easy access to scientific data and data products
- Scientific databases grow in size (to TB-scale and beyond), which poses growing challenges on their performance [Gray and Szalay, 2002]
- Performance enhancements are most effective if the database and connected applications / (web-) services are tuned together, in a context-aware approach [Nimalasena and Getov, 2014]

Example screenshots from JOIN web interface to TOAR database

TOAR-DB service infrastructure

Provides global air quality metrics according to established standards

- The TOAR – DB is one of the world's largest databases for near-surface air quality measurements and base for TOAR peer reviewed papers [Schultz et al., 2017; Chang et al., 2017; Fleming et al., 2018; Gaudel et al., 2018; Lefohn et al., 2018; Mills et al., 2019; Young et al., 2019; Tarasick et al., 2019; Xu et al., 2020]
- The TOAR-DB comprises terabytes of measurements collected from public bodies, research institutions and air quality networks all over the world. It thus enables global assessment of air quality measurements
- Special feature of TOAR-DB service infrastructure: On-demand calculation of well documented air quality metrics. Data and air quality metrics are openly accessible via the graphical web interface JOIN or a REST API on HDF cloud [Hagemeier, 2019]
- Description of current set-up (see image below): **1)** The end user requests statistical quantities **2)** the web services trigger SQL queries via the Python psycopg2 library **3)** the resulting raw data are transferred to the JOIN server **4)** The data is processed on the JOIN server, then products are provided to the end user

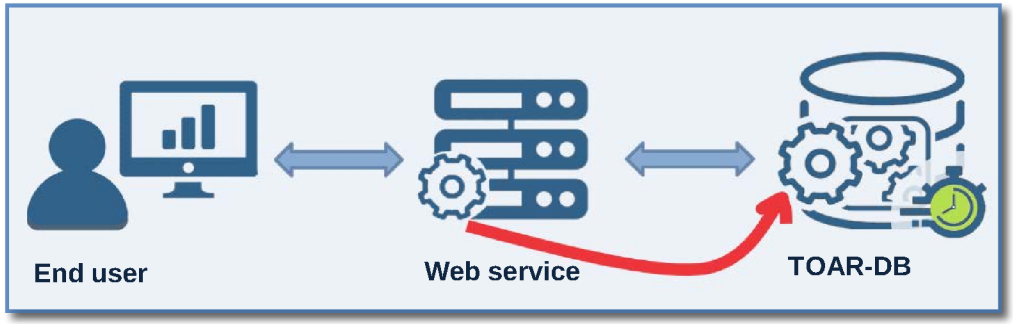


Set-up of TOAR-DB service infrastructure

TOAR-DB service infrastructure

Provides global air quality metrics according to established standards

- The TOAR - DB is one of the world's largest databases for near-surface air quality measurements and base for TOAR peer reviewed papers [Schultz et al., 2017; Chang et al., 2017; Fleming et al., 2018; Gaudel et al., 2018; Lefohn et al., 2018; Mills et al., 2019; Young et al., 2019; Tarasick et al., 2019; Xu et al., 2020]
- The TOAR-DB comprises terabytes of measurements collected from public bodies, research institutions and air quality networks all over the world. It thus enables global assessment of air quality measurements
- Special feature of TOAR-DB service infrastructure: On-demand calculation of well documented air quality metrics. Data and air quality metrics are openly accessible via the graphical web interface JOIN or a REST API on HDF cloud [Hagemeier, 2019]
- Description of current set-up (see image below): **1)** The end user requests statistical quantities **2)** the web services trigger SQL queries via the Python psycopg2 library **3)** the resulting raw data are transferred to the JOIN server **4)** The data is processed locally, then provided to the end user



Set-up of TOAR-DB service infrastructure

Motivation for this work

- Throughput of our services is limited!
- Enhance by data processing **inside** the DB (as indicated in the image)?
- Design benchmarks of different tasks and test cases to check performance aspects individually

Performance analysis and optimization of a TByte-scale atmospheric observation database

Introduction: Fast access to scientific data products

- Performance of scientific database applications
- TOAR-DB service infrastructure

TOAR-DB performance benchmarks

- Overview
- Data aggregation inside versus outside the database
- Metrics calculation inside versus outside the database
- Parallel processing
- Influence of indices on query times
- Transfer times between database server and web server

Extend the functionality: Modeling global stomatal ozone fluxes

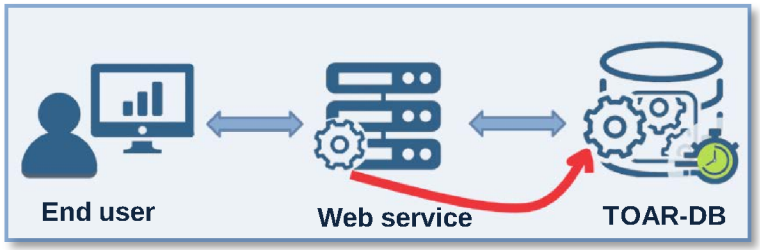
- Motivation: Ozone impact on vegetation
- Resources and realization

Summary and conclusions

Overview

TOAR-DB Performance benchmarks

- Main objective: Data processing inside the database via SQL user defined functions (indicated by red arrow in figure)
- Design different benchmarks, each highlighting different performance aspects and use cases
- Each benchmark consists of different tasks and test cases (see table below, elaboration follows)



Set-up of TOAR-DB service infrastructure

- Focus on queries and processing of TOAR-DB ozone table. It has 10^9 rows, each containing timeseries_identifier, timestamp, value, status flag. Primary key is on first two columns

Table: Benchmark overview

Benchmark	Tasks	Test cases
Data aggregation inside versus outside the database	Count, maximum, average, standard deviation of ozone values for given dates	1) "Python" 2) "SQL" 3) "PL/Pythonu"
Metrics calculation inside versus outside the database	Drmdmax1h, AOT40, dma8epa, W90 for given series ids and years	1) "Python" 2) "SQL"
Parallel processing	Parallel scan, Parallel aggregate	1) max. 1 worker 2) max. 2 workers 3) max. 4 workers 4) max. 8 workers
Influence of **indices** on query times	Maximum value for a given date (SQL query)	1) "o3_hourly" 2) "temp_hourly"
Transfer times between database server and web server	Test bandwidths and latency	1) Ping round trip time test 2) Bandwiths test with iperf3

Data aggregation inside vs. outside the database

Motivation and tasks

- The performance of simple aggregates generalizes to more complex applications

- Here, we picked a random date and aggregated all ozone values from this date

- Tasks: counting entries (‘count’), finding the maximum entry (‘max’), the mean value of all entries (‘avg’) and their standard deviation (‘std’)

- Transfer times and caching were avoided

Test cases

- “Python”: The temporally filtered data was queried from the database and further aggregated in the Python data science framework NumPy (use case so far)

- “SQL”: The data was filtered and aggregated in form of an SQL query, then loaded into the Python cache

- “PL/Pythonu”: The aggregate was calculated inside the database by a user defined function in the imported procedural language PL/Pythonu, then loaded into the Python cache

Results

- Benchmark results. The cells contain execution time and standard deviation (n=100 each)

Aggregate	“Python” μ±σ [s]	“SQL” μ±σ [s]	“PL/ Pythonu“ μ±σ [s]	Rel. Diff. “Python”- “SQL”
‘count’	0.23 ± 0.05	0.18 ± 0.08	0.22 ± 0.03	21.7 %
‘max’	0.27 ± 0.05	0.21 ± 0.07	0.26 ± 0.07	22.2 %
‘avg’	0.25 ± 0.04	0.17 ± 0.03	0.26 ± 0.03	32.0 %
‘std’	0.23 ± 0.04	0.16 ± 0.02	0.27 ± 0.06	30.4 %

Discussion

➡ **SQL aggregates are the fastest**, with average time savings of 21.7% to 32.0%

➡ When comparing Python and SQL (last column in table), **less** time is saved with less computationally complex calculations (‘count’) and **more** time is saved with more complex the calculation (‘avg’, ‘std’)

Air quality metrics calculation inside vs. outside the database

Motivation and tasks

- Main performance-critical use cases: calculation of different yearly air quality metrics on the ozone data table inside vs. outside the database

- Four common metrics were considered: 'drmdmax1h', 'AOT40', 'dma8epa', 'W90' (Calculations require combination of (rolling) aggregation and data capture filtering)

- Metrics are documented by Schultz et al. [2017]

Test cases

- "Python": Hourly ozone data from the given time series and period was queried from the database, and further processed in the Python data science framework Pandas (use case so far)

- "SQL": The calculation was rewritten in a user defined function in SQL, and performed inside the database. Then the end result was loaded into the Python cache

Results

- Benchmark results. The cells contain execution time and standard deviation (n=250 each).

Metric	Python μ±σ [s]	SQL μ±σ [s]	Relative difference
'drmdmax1h'	0.18 ± 0.01	0.15 ± 0.01	16.7 %
'AOT40'	0.19 ± 0.02	0.15 ± 0.02	21.1 %
'dma8epa'	0.18 ± 0.02	0.17 ± 0.02	5.6 %
'W90'	0.18 ± 0.02	0.17 ± 0.02	5.6 %

Discussion

- **Calculating metrics in SQL is faster than Python**. The average time difference varies between 5.6% and 21.1%

- In SQL, metrics that include the calculation of hourly **rolling means/sums** ('W90', 'dma8epa') are comparatively slower than metrics that include only **aggregates** ('drmdmax1h', 'AOT40')

- SQL query plan: 50% of the time spent with temporal filtering, 50% for processing

Parallel processing

Motivation and tasks

- In contrast to the two previous benchmarks, here we examine database speedup via parallel scans and aggregation in the database

- Tasks allow effective parallel processing. ‘scan’: parallel index scan across the entire ozone table that filters all ozone values below zero, indicating incorrect values. ‘agg’: parallel scan and aggregate across the full ozone table to output the mean of all ozone values

Test cases

- The speedup is tested by varying the maximum number of parallel workers allowed in a query

- Allowed number of workers: 1, 2, 4, 8

Results

- Benchmark results. The cells contain the actual number of workers spawned, the execution time, and the difference to execution time with one worker.

Task	1 worker allowed [#] / [s] / [%]	2 workers allowed [#] / [s] / [%]	4 workers allowed [#] / [s] / [%]	8 workers allowed [#] / [s] / [%]
‘scan’	1 0.93	2 0.68 - 26.9 %	4 0.65 - 30.1 %	5 0.98 + 5.4%
‘agg’	1 99.03	2 78.71 - 20.5 %	4 71.43 - 27,9 %	8 76.10 - 23.2 %

Discussion

➡ The query planner considers time needed to **spawn** workers, **process** in parallel and **gather** results

➡ Depending on these times it may be infeasible to spawn all workers which are allowed (see ‘scan’ with 8 workers allowed but only 5 workers spawned, which takes longer than 4 workers)

➡ Parallel processing with more than 2 workers did not speed up the process. CPU load was always < 25%. This shows that the **system is I/O bound**.

Influence of indices on query times

Motivation and task

- Air quality monitoring data consists of time series, so many database queries require filtering over time

- Here we examine the importance of indices for the performance of temporal filtering in the database

- The task corresponds aggregate benchmark (Slide 10), test case “SQL”: query all data from a randomly selected date, and output the maximum value

Test cases

- Compare performance tables that are similar in size and structure, but have different indices

- First, the ‘o3_hourly’ ozone table with index on datetime, value and id in addition to the primary key

- Second, the ‘temp_hourly’ table, which contains hourly temperature values and has only the primary key on id and datetime

Results

- Benchmark results

Aggregate	“o3_hourly” (n=100) μ±σ [s]	“temp_hourly” (n=20) μ±σ [s]
‘max SQL’	0.21 ± 0.07	82.63 ± 2.41

Discussion

- Temporal filtering took **several magnitudes longer** without the index on datetime, id and value
- This also shows that the order of the indices is crucial
- Depending on whether a **time range** or a **time series** is of interest, different indices are used

Transfer times between database server and web server

Motivation and task

- DB server and web server are virtual machines on different hosts in HDF cloud [Hagemeier, 2019]
- This means that transfer between machines is not avoidable
- Does 'in-DB' processing also save time by avoiding transfer between machines? If yes, how much?
- Test bandwidths and latency of transfer between machines

Test cases

- Test latency between virtual machines via the ping round trip time
- Test bandwidths using the iperf3 tool

Results

Transfer	bandwidths	latency
Between virtual machines in HDF cloud	8.3 Gbit/s	0.7 ms ping round trip time

Discussion

➡ **Ping round trip time is not avoidable** regardless the amount of data transferred

➡ When comparing in-DB processing (transfer negligible) vs. web server processing (hourly data of 10 years = ca. 1.2 MB tansferred), we notice that the **transfer time is < 1 ms and thus negligible**

Transfer times between database server and web server

Motivation and task

- DB server and web server are virtual machines on different hosts in HDF cloud [Hagemeier, 2019]
- This means that transfer between machines is not avoidable
- Does 'in-DB' processing also save time by avoiding transfer between machines? If yes, how much?
- Test bandwidths and latency of transfer between machines

Test cases

- Test latency between virtual machines via the ping round trip time
- Test bandwidths using the iperf3 tool

Results

Transfer	bandwidths	latency
Between virtual machines in HDF cloud	8.3 Gbit/s	0.7 ms ping round trip time

Discussion

➡ **Ping round trip tim**
the amount of data trans

➡ When comparing in-
negligible) vs. web serve
years = ca. 1.2 MB tans
transfer time is < 1 ms

"So much for the benchmarks. The final conclusions come on last slide. But first, let us take a look at what else you can do with this Web - Service set up!"

Performance analysis and optimization of a TByte-scale atmospheric observation database

Introduction: Fast access to scientific data products

- Performance of scientific database applications
- TOAR-DB service infrastructure

TOAR-DB performance benchmarks

- Overview
- Data aggregation inside versus outside the database
- Metrics calculation inside versus outside the database
- Parallel processing
- Influence of indices on query times
- Transfer times between database server and web server

Extend the functionality: Modeling global stomatal ozone fluxes

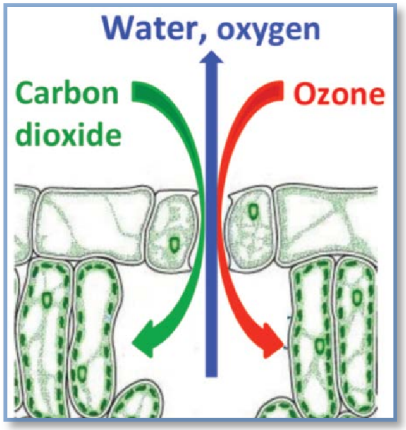
- Motivation: Ozone impact on vegetation
- Resources and realization

Summary and conclusions

Motivation: Ozone impact on vegetation

We have a great DB + Web service → What else can we do with it?

- Ozone harms crops and vegetation (see images). It is important to quantify the damage
- So far, TOAR + JOIN enabled global assessment of damage only by concentration based metrics [Mills et al., 2018]
- But: **Ozone impact on vegetation can be quantified much better if the plant growing conditions are taken into account! → Flux based ozone metrics** [Mills et al., 2011]
- Flux-based ozone metrics need **1)** input of meteorological data **2)** consistent parametrization of plant phenology **3)** integrated flux modeling with the DO$_3$SE model



Cross section of leaf pore ('stomata'). Picture adapted from ICP brochure "Flux-based critical levels of ozone pollution for vegetation"

Visible damage on French bean leaves. Picture adapted from: ICP brochure "Have you seen these ozone injury symptoms?"

Resources and realization

TOAR + DO_3SE = Web - Service

→ Global, flux-based vegetation damage assessment possible

How can we bring together TOAR-DB and DO_3SE model?

- **Meteorological data:**
  - Extraction from gridded model data to air quality measurement station locations
  - Stored either in the database or on the web server
- **Phenology parametrization:**
  - The plant ecophysiologist community has to agree on parametrizations of growing seasons and phenology
  - This is planned for the next phase of TOAR: TOAR-II
- **DO3SE model:**
  - The model is written in FORTRAN
  - It can be compiled and linked to the Python web-service via the F2PY module
- **Further use:**
  - TOAR-II vegetation paper (planned)
  - Assessment by International Cooperative Programme on Effects of Air Pollution on Natural Vegetation and Crops (ICP)

Performance analysis and optimization of a TByte-scale atmospheric observation database

Introduction: Fast access to scientific data products

- Performance of scientific database applications
- TOAR-DB service infrastructure

TOAR-DB performance benchmarks

- Overview
- Data aggregation inside versus outside the database
- Metrics calculation inside versus outside the database
- Parallel processing
- Influence of indices on query times
- Transfer times between database server and web server

Extend the functionality: Modeling global stomatal ozone fluxes

- Motivation: Ozone impact on vegetation
- Resources and realization

Summary and conclusions

Summary and conclusions

Fast access to scientific data products

- ✓ Increases the quality, flexibility, reproducibility and outreach of scientific workflows
- ✓ Performance issues grow with size of DB
- ✓ Context – aware tuning of DB and service infrastructure recommended

TOAR-DB Performance benchmarks

- ✓ Server-side programming in SQL increases performance by up to 32%
- ✓ Optimal definition of indices is crucial, depending if filtered by time or series
- ✓ Data transfer times between web service and DB are negligible
- ✓ DB performance is I/O bound, so parallel scanning and processing does not scale well

Extend the functionality: Modeling global stomatal ozone fluxes

- ✓ TOAR service infrastructure can be extended with embedded flux modeling
- ✓ This will allow a global assessment of flux based ozone metrics for the first time

Thanks for your attention!

Acknowledgements

ESDE group at Jülich Supercomputing Centre (@ home office) …

Björn Hagemeier

… and (co -) authors of this display

Literature

- Chang, K.-L., Petropavlovskikh, I., Copper, O.R., Schultz, M.G., Wang, T., 2017. Regional trend analysis of surface ozone observations from monitoring networks in eastern North America, Europe and East Asia. Elem Sci Anth 5, 50. https://doi.org/10.1525/elementa.243
- Fleming, Z.L., Doherty, R.M., Von Schneidemesser, E., Malley, C.S., Cooper, O.R., Pinto, J.P., Colette, A., Xu, X., Simpson, D., Schultz, M.G., Lefohn, A.S., Hamad, S., Moolla, R., Solberg, S., Feng, Z., 2018. Tropospheric Ozone Assessment Report: Present-day ozone distribution and trends relevant to human health. Elem Sci Anth 6, 12. https://doi.org/10.1525/elementa.273
- Gaudel, A., Cooper, O.R., Ancellet, G., Barret, B., Boynard, A, et al., 2018. Tropospheric Ozone Assessment Report: Present-day distribution and trends of tropospheric ozone relevant to climate and global atmospheric chemistry model evaluation. Elem Sci Anth 6, 39. https://doi.org/10.1525/elementa.291
- Gray, J., Szalay, A., 2002. The world-wide telescope. Communications of the ACM 45, 50–55. https://doi.org/10.1145/581571.581572
- Hagemeier, B., 2019. HDF Cloud – Helmholtz Data Federation Cloud Resources at the Jülich Supercomputing Centre. Journal of large-scale research facilities JLSRF 5. https://doi.org/10.17815/jlsrf-5-173
- Lefohn, A.S., Malley, C.S., Smith, L., Wells, B., Hazucha, M., et al., 2018. Tropospheric ozone assessment report: Global ozone metrics for climate change, human health, and crop/ecosystem research. Elem Sci Anth 6, 28. https://doi.org/10.1525/elementa.279
- Mills, G., Hayes, F., Simpson, D., Emberson, L., Norris, D., et al., 2011. Evidence of widespread effects of ozone on crops and (semi-) natural vegetation in Europe (1990-2006) in relation to AOT40- and flux-based risk maps: OZONE EFFECTS ON VEGETATION IN EUROPE. Global Change Biology 17, 592–613. https://doi.org/10.1111/j.1365-2486.2010.02217.x
- Mills, G., Pleijel, H., Malley, C.S., Sinha, B., Cooper, O.R., et al., 2018. Tropospheric Ozone Assessment Report: Present-day tropospheric ozone distribution and trends relevant to vegetation. Elem Sci Anth 6, 47. https://doi.org/10.1525/elementa.302
- Nimalasena, A., Getov, V., 2014. Performance tuning of database systems using a context-aware approach, in: 2014 9th International Conference on Computer Engineering & Systems (ICCES). Presented at the 2014 9th International Conference on Computer Engineering & Systems (ICCES), IEEE, Cairo, Egypt, pp. 98–103. https://doi.org/10.1109/ICCES.2014.7030936
- Schultz, M.G., Schröder, S., Lyapina, O., Cooper, O., Galbally, I., et al., 2017. Tropospheric Ozone Assessment Report: Database and Metrics Data of Global Surface Ozone Observations. Elem Sci Anth 5, 58. https://doi.org/10.1525/elementa.244
- Tarasick, D., Galbally, I.E., Cooper, O.R., Schultz, M.G., Ancellet, G., et al., 2019. Tropospheric Ozone Assessment Report: Tropospheric ozone from 1877 to 2016, observed levels, trends and uncertainties. Elem Sci Anth 7, 39. https://doi.org/10.1525/elementa.376
- Xu, X., 2018. Tropospheric Ozone Assessment Report: Long-term changes of regional ozone in China: implications for human health and ecosystem impacts Elem Sci Anth 7, 13. https://doi.org/10.1525/elementa.409
- Young, P.J., Naik, V., Fiore, A.M., Gaudel, A., Guo, J., et al., 2018. Tropospheric Ozone Assessment Report: Assessment of global-scale model performance for global and regional ozone distributions, variability, and trends. Elem Sci Anth 6, 10. https://doi.org/10.1525/elementa.265