Serverless Speed: Rust vs. Go, Java, and Python in AWS Lambda Functions

Serverless Speed: Rust vs. Go, Java, and Python in AWS Lambda Functions

- https://blog.scanner.dev/serverless-speed-rust-vs-go-java-python-in-aws-lambda-functions/

Takeaways

- Use 1.5GB+ memory allocation for best S3 thruput
- Benchmark JSON libraries

Rust Lambda setup

```
// main.rs
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use rusoto_core::Region;
use rusoto_s3::{S3Client, S3};
use serde::{Deserialize, Serialize};
use tokio::io::{AsyncReadExt, AsyncBufReadExt};

#[derive(Deserialize)]
struct Request {
    bucket: String,
    key: String,
}
```

```
#[derive(Serialize)]
struct Response {
    req_id: String,
    msg: String,
}

async fn handle_request(event: LambdaEvent<Request>) -> Result<Response, Error> {
    let started_at = std::time::Instant::now();
    let client = S3Client::new(Region::UsWest2);

    let output = client
        .get_object(rusoto_s3::GetObjectRequest {
            bucket: bucket.to_string(),
            key: key.to_string(),
            ..Default::default()
        })
        .await?;

    let Some(body) = output.body else {
        return Err(anyhow::anyhow!("No body found in S3 response").into());
    };

    let body = body.into_async_read();
    let body = tokio::io::BufReader::new(body);
    let decoder = async_compression::tokio::bufread::ZstdDecoder::new(body);
    let reader = tokio::io::BufReader::new(decoder);

    let mut lines = reader.lines();
    let mut num_log_events = 0;
    while let Ok(Some(mut line)) = lines.next_line().await {
        let _value = unsafe {
            simd_json::to_borrowed_value(line.as_mut_str().as_bytes_mut())?
        };
        num_log_events += 1;
        if num_log_events % 1000 == 0 {
            println!("num_log_events={}", num_log_events);
        }
    }

    let msg = format!(
```

```
        "elapsed={:?} num_log_events={}",
        started_at.elapsed(),
        num_log_events
    );
    Ok(Response {
        req_id: event.context.request_id,
        msg,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .init();

    run(service_fn(handle_request)).await
}
```

Build:

```
# build.sh
cargo lambda build --release --arm64
(cd ./target/lambda/lambda_langs_test_rust/ && zip ./bootstrap.zip ./bootstrap)
```

Deploy:

```
aws lambda create-function \
--function-name lambda_langs_test_rust \
--runtime provided.al2 \
--memory-size 640  \
--architectures arm64 \
--zip-file ./bootstrap.zip \
--handler unused \
--timeout 900 \
--role ${LAMBDA_IAM_ROLE}
```

Run:

```
aws lambda invoke \
  --function-name lambda_langs_test_python \
```

```
  --log-type Tail \
  --cli-binary-format raw-in-base64-out \
  --payload '{"bucket": "<s3_bucket>", "key": "<s3_key>"}' \
  ./response.json \
| jq -r .LogResult | base64 --decode
```

Revision #1
Created 2023-04-21 09:58:51 IST by hxd
Updated 2023-04-21 10:05:00 IST by hxd