International
Journal
Of Advanced Research in Engineering & Management (IJAREM)
ISSN: 2456-2033 || PP. 01-11

Method for Verifying the Functional Equivalence of Server Energy Efficiency Benchmark Software Tools

Anders S. G. Andrae
Huawei Technologies Sweden AB, Sweden

Abstract: Servers are essential for data centers including their electricity use. It is important to be able to measure the energy efficiency (EE) of servers to keep track of the power consumption of data centers. Various so-called EE benchmarking software tools are used for standardizing the measurement. It is yet unknown if these tools are functionally equivalent, i.e. ranking and calibration equivalent. Here is presented a pioneering implementation for testing the functional equivalence of server EE measurement tools. For the first time, Benchmark of Server Energy Efficiency (BenchSEE) and Server Efficiency Rating Tool (SERT) are tentatively compared in terms of ranking and calibration equivalence. The preliminary suggestion is that the EE benchmarking tools BenchSEE and SERT are functionally interchangeable based on ranking and calibration equivalence. However, the research provides only tentative indications, limited by the small sample size.

Keywords: benchmark, calibration, energy efficiency, ranking, servers, tools

I. Introduction

The data centers are using ever more power. [1] The largest user of power in a data center is the IT equipment like servers. [2]Therefore it is crucial to be able to measure the server power consumption from different throughputs, i.e. the energy efficiency. Research featuring energy efficiency (EE) benchmark tools such as Benchmark of Server Energy Efficiency (BenchSEE) and Server Efficiency Rating Tool (SERT) [3]is getting more attention. [4]-[13] This is done to find strategies for power savingfor server and supercomputer design. In this context a standard for EE assessment of heterogeneous servers was recently released. [14]

The research gap in the literature is that so far that the equivalence features of BenchSEE and SERT have not been investigated.

So far there is no research analyzing if BenchSEE and SERT are numerically linear consistent, i.e. calibration equivalent. If possessing such equivalence, their results would be the same up to a fixed predictable conversion with a certain error tolerance. After the conversion is applied the two tools would lineup closely across all cases. The same ranking (Spearman coefficient ≈1) would also be preserved.

An analogy is temperature measurement with Thermometer Y expressed in Fahrenheit and Thermometer Z expressed in Celsius. It is obviously possible to use Thermometers Y and Z for the same measurement and compare the results. In this case Thermometers Y and Z are functionally equivalent. Multiple other analogies can be found [15] in which different tools express results in different units while (the tools) still being ranking and calibration equivalent. The various unit systems for energy, mass, temperature etc. and corresponding measurement tools offer such cases. [16]For instance, energy meters could be tested for functional equivalence. [17],[18]

The objective of this research is to investigate if BenchSEE and SERT could be ranking and calibration equivalent with a tolerance error of <10%, i.e. determine if the tools are functionally equivalent.

In this investigation the tools have ranking equivalence if the Spearman coefficient $\rho \geq 0.99$, and they have scoring/calibration equivalence if the

- Mean Absolute Percentage Error (MAPE) ≤ 10%
- Max |% error| ≤ 10% across points.

For the first time, BenchSEE and SERT are tentatively compared in terms of ranking and calibration equivalence. The present research is based on literature. [7] Therefore the present study will only offer an initial suggestion.

II. Theoretical Framework

The implementation is based on data pairs for EE benchmark scores. EE benchmarks are based on worklets. A worklet is a small, focused mini-work which acts on a specific part of the server (e.g. Central PU, memory etc) at several different loads for which both the throughput and the power use are measured. Worklets are building blocks run in series, combined to larger workloads and the results are presented as EE benchmarks.

As shown in Table 1 eleven different worklets are used.

Table 1: Worklet types included in BenchSEE and SERT

Worklets	BenchSEE name (values from author of [4])	SERT name (values from author of [4])
CPU related 1	Active Ide Efficiency Score (AES)	CryptoAES
CPU related 2	Compress	Compress
CPU related 3	Load Unit (LU)	LU
CPU related 4	Sort	Sort
CPU related 5	Secure Hash Algorithm 256 (SHA256)	SHA256
CPU related 6	SORting (SOR)	SOR
CPU related 7	Online Transaction Processing (OLTP)	Server Side Java (SSJ)
Memory related 1	Cache	Capacity
Memory related 2	Stream	Flood
Storage related 1	Random	Random
Storage related 1	Sequential	Sequential

On request from the authors of [7], the throughput, the power use of the worklets used for the server used, and the resulting absolute *EE* values for "Default" (blue bars) in Fig. 5 and Fig. 6 in [7] were obtained facilitating the calculation of SERT1 and BenchSEE1. Then the *EE* values for the second series are estimated from the green bars, the third series are estimated from the orange bars and the ones for the fourth series are based on the red bars in Fig. 5 and Fig. 6 in [7].

A. Calculation of the EE Benchmark

The *EE* benchmarks are calculated according to (1), (2), (3) and (4) which use the values from Table 2.

$$EE_{CPU} = \left(\prod_{k=1}^{n_{CPU}} EE_{CPU,k}\right)^{1/n_{CPU}} \quad (1)$$

$$EE_{memory} = \left(\prod_{i=1}^{n_{mem}} EE_{Mem,i}\right)^{1/n_{mem}} \quad (2)$$

$$EE_{storage} = \left(\prod_{j=1}^{n_{sto}} EE_{Sto,j}\right)^{1/n_{sto}} \quad (3)$$

$$EE_{total} = e^{((0.65\times\ln(EE_{CPU})+(0.3\times\ln(EE_{memory})+(0.05\times ln(EE_{storage}))} \quad (4)$$

Table 2 shows all of the EE_{CPU} ,EE_{memory} and $EE_{storage}$ values calculated and used by (1), (2), (3) and (4)**.**

Table 2: Basis for ranking and regression calculations

Parameter	SERT 1	SERT 2	SERT 3	SERT 4	BenchSEE 1	BenchSEE 2	BenchSEE 3	BenchSEE 4
EE_{CPU}	242.79	302.53	322.51	331.44	275.93	291.77	343.95	354.61
EE_{memory}	58.20	61.34	68.54	69.07	116.17	111.42	120.76	121.17
$EE_{storage}$	0.81	0.78	1.04	1.02	8.02	7.86	10.42	10.82
EE_{total}	**118.96**	**139.10**	**152.10**	**155.02**	**178.34**	**182.44**	**210.96**	**215.81**

The EE_{total} values Table 2 are then used to calculate the statistical properties of the system.

These statistical properties are evaluated with the following equations 5 to 15 in which *A* represents BenchSEE data and *B* represents SERT data:

$$B_i = \alpha A_i + \beta. \tag{5}$$

$$X = \begin{bmatrix} \vdots & \vdots \\ 1 & A_n \end{bmatrix}, \theta = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{6), (7}$$

$$\theta = (X^\top X)^{-1} X^\top B. \tag{8}$$

$$\alpha = \frac{\sum_i (A_i - \bar{A})(B_i - \bar{B})}{\sum_i (A_i - \bar{A})^2} \tag{9}$$

$$\beta = \bar{B} - \alpha\, \bar{A}. \tag{10}$$

$$\widehat{B}_i = \alpha A_i + \beta \tag{11}$$

$$e_i = B_i - \widehat{B}_i \tag{12}$$

$$R^2 = 1 - \frac{\sum_i (B_i - \widehat{B}_i)^2}{\sum_i (B_i - \bar{B})^2}. \tag{13}$$

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{B_i - \widehat{B}_i}{B_i} \right|. \tag{14}$$

$$\rho_s = \frac{\sum_i (R_{A,i} - \overline{R_A})(R_{B,i} - \overline{R_B})}{\sqrt{\sum_i (R_{A,i} - \overline{R_A})^2} \sqrt{\sum_i (R_{B,i} - \overline{R_B})^2}} \tag{15}$$

where

A_i: Value from **tool A** for case i (independent variable).

B_i: Value from **tool B** for case i (dependent variable).

α: **Slope** of the affine calibration line $B = \alpha A + \beta$ (scale factor).

β: **Intercept** of the affine calibration line (offset/bias).

$\widehat{B}_i$: **Predicted** value of B_i from the fitted line, $\widehat{B}_i = \alpha A_i + \beta$.

e_i: **Residual** (prediction error) for case i, $e_i = B_i - \widehat{B}_i$.

$\bar{A}$: **Mean** of all A_i.

$\bar{B}$: **Mean** of all B_i.

n: Number of paired observations (A_i, B_i).

R^2: **Coefficient of determination** (fraction of variance in B explained by the fitted line).

MAPE: **Mean Absolute Percentage Error**, average $\left|(B_i - \widehat{B}_i)/B_i\right|$ in percent.

$\%\mathrm{err}_i$: **Percent error** for case i, $\left|(B_i - \widehat{B}_i)/B_i\right| \times 100\%$.

$\max|\%\mathrm{err}|$: **Worst-case percent error** across all points.

ρ_s: **Spearman's rank correlation** between A and B (correlation of their ranks).

$R_{A,i}$: **Rank** of A_i among all A-values (average rank if ties).

$R_{B,i}$: **Rank** of B_i among all B-values (average rank if ties).

X: **Design matrix** $\begin{bmatrix} 1 & A_1 \\ \vdots & \vdots \\ 1 & A_n \end{bmatrix}$.

θ: Parameter vector $\begin{bmatrix} \beta \\ \alpha \end{bmatrix}$.

SS_{res}: Residual sum of squares $\sum(B_i - \widehat{B}_i)^2$.

SS_{tot}: Total sum of squares $\sum(B_i - \bar{B})^2$.

B. Code in GNU Octave for ranking and calibration equivalence investigation

The following codesare used in GNU Octave [19] to generate Fig. 1, Fig. 2 and Fig. 3, respectively.The codes are presented for full transparency and repeatability of the present research.

1) Code for Fig. 1

```
% Data (A: BenchSEEindependent, B: SERT dependent)
A = [178.34; 182.44; 210.96; 215.81];
B = [118.96; 139.1; 152.1; 155.02];

% Design matrix (intercept + slope)
X = [ones(size(A)), A];

% Least squares (numerically stable)
theta = X \ B;              % same as (X'*X)\(X'*B) but better conditioned
beta  = theta(1);           % intercept
alpha = theta(2);           % slope
```

```
% Predictions and residuals
B_hat = X * theta;
res   = B - B_hat;

% Per-point percent errors and max |% error|
pct_err = abs((B - B_hat)./B) * 100;     % [%]
max_pct_err = max(pct_err);

% Fit quality
SS_tot = sum((B - mean(B)).^2);
SS_res = sum(res.^2);
R2     = 1 - SS_res/SS_tot;

% Error metrics
MAPE = mean(abs(res)./abs(B))*100; % mean absolute percentage error

% ---------- Spearman rank correlation (no toolboxes needed) ----------
% Try built-in corr if available; otherwise use fallback function.
if exist('corr','file')
  try
    rho = corr(A, B, 'type', 'Spearman');
  catch
    rho = spearman_rho(A, B);
  end
else
  rho = spearman_rho(A, B);
end
% ---------------------------------------------------------------------

% Display in console
fprintf('alpha (slope)   = %.4f\n', alpha);
fprintf('beta (intercept)= %+.4f\n', beta);
fprintf('R^2             = %.4f\n', R2);
fprintf('MAPE            = %.2f %%\n', MAPE);
fprintf('Max |%% error|  = %.2f %%\n', max_pct_err);
fprintf('Spearman rho    = %.4f\n', rho);

% Plot data + regression line + annotation
figure; hold on; grid on;
plot(A, B, 'bo', 'MarkerSize', 30, 'DisplayName', 'Data');

A_line = linspace(min(A), max(A), 200);
B_line = beta + alpha*A_line;
plot(A_line, B_line, 'r-', 'LineWidth', 2, 'DisplayName', 'B = \alpha A + \beta');

legend('Location','southeast');
xlabel('A'); ylabel('B');
title(sprintf('Linear Fit (R^2 = %.4f)', R2));

% Put equation, errors & Spearman on the plot
fmt = ['B = %.4f·A %+.4f\n' ...
       'MAPE = %.2f%%\n' ...
       'Max |%% err| = %.2f%%\n' ...
       'Spearman \xCF\x81 = %.3f'];   % \xCF\x81 = Greek rho in TeX text
txt = sprintf(fmt, alpha, beta, MAPE, max_pct_err, rho);

xpos = min(A) + 0.05*(max(A)-min(A));
ypos = max(B) - 0.05*(max(B)-min(B));
text(xpos, ypos, txt, ...
    'VerticalAlignment','top', ...
    'FontName','monospace', ...
    'FontSize', 35, ...
    'FontWeight','bold', ...
    'BackgroundColor','w', ...
    'Margin',6, ...
    'Interpreter','tex');   % ensures Greek rho renders

% (MATLAB) global-ish defaults for this session:
set(groot,'DefaultAxesFontSize',24);
set(groot,'DefaultTextFontSize',24);

% (Octave-compatible) use the root 0:
set(0,'DefaultAxesFontSize',24);
set(0,'DefaultTextFontSize',24);

% Per-object sizing:
xlabel('BenchSEE(A)','FontSize',50,'FontWeight','bold');
ylabel('SERT (B)','FontSize',50,'FontWeight','bold');
title(sprintf('Linear Fit (R^2 = %.4f)', R2), 'FontSize', 50, 'FontWeight','bold');
legend('Location','southeast','FontSize',50);
set(gca,'FontSize',50,'LineWidth',1.25);
set(gcf,'Color','w','Units','normalized','Position',[0.1 0.1 0.6 0.6]);

% ==================== helper functions ====================
function r = spearman_rho(x,y)
  x = x(:); y = y(:);
  ok = ~(isnan(x) | isnan(y));
  x = x(ok); y = y(ok);
rx = avg_tied_ranks(x);
ry = avg_tied_ranks(y);
rx = rx - mean(rx); ry = ry - mean(ry);
  r = (rx' * ry) / sqrt((rx' * rx) * (ry' * ry));
```

```
end

function r = avg_tied_ranks(v)
  [sv, idx] = sort(v);
  n = numel(v); r = zeros(n,1);
  i = 1;
  while i <= n
    j = i;
    while j <= n &&sv(j) == sv(i), j = j + 1; end
rank_ij = (i + j - 1) / 2;            % average rank for ties
    r(idx(i:j-1)) = rank_ij;
    i = j;
  end
end
```

2) Code for Fig. 2

```
% Data (A: BenchSEEindependent, B: SERT dependent)
A = [178.34; 210.96; 215.81];
B = [118.96; 152.1; 155.02];

% Design matrix (intercept + slope)
X = [ones(size(A)), A];

% Least squares (numerically stable)
theta = X \ B;                % same as (X'*X)\(X'*B) but better conditioned
beta  = theta(1);             % intercept
alpha = theta(2);             % slope

% Predictions and residuals
B_hat = X * theta;
res   = B - B_hat;

% Per-point percent errors and max |% error|
pct_err = abs((B - B_hat)./B) * 100;     % [%]
max_pct_err = max(pct_err);

% Fit quality
SS_tot = sum((B - mean(B)).^2);
SS_res = sum(res.^2);
R2     = 1 - SS_res/SS_tot;

% Error metrics
MAPE = mean(abs(res)./abs(B))*100; % mean absolute percentage error

% ---------- Spearman rank correlation (no toolboxes needed) ----------
% Try built-in corr if available; otherwise use fallback function.
if exist('corr','file')
  try
    rho = corr(A, B, 'type', 'Spearman');
  catch
    rho = spearman_rho(A, B);
  end
else
  rho = spearman_rho(A, B);
end
% ---------------------------------------------------------------------

% Display in console
fprintf('alpha (slope)   = %.4f\n', alpha);
fprintf('beta (intercept)= %+.4f\n', beta);
fprintf('R^2             = %.4f\n', R2);
fprintf('MAPE            = %.2f %%\n', MAPE);
fprintf('Max |%% error|  = %.2f %%\n', max_pct_err);
fprintf('Spearman rho    = %.4f\n', rho);

% Plot data + regression line + annotation
figure; hold on; grid on;
plot(A, B, 'bo', 'MarkerSize', 30, 'DisplayName', 'Data');

A_line = linspace(min(A), max(A), 200);
B_line = beta + alpha*A_line;
plot(A_line, B_line, 'r-', 'LineWidth', 2, 'DisplayName', 'B = \alpha A + \beta');

legend('Location','southeast');
xlabel('A'); ylabel('B');
title(sprintf('Linear Fit (R^2 = %.4f)', R2));

% Put equation, errors & Spearman on the plot
fmt = ['B = %.4f·A %+.4f\n' ...
       'MAPE = %.2f%%\n' ...
       'Max |%% err| = %.2f%%\n' ...
       'Spearman \xCF\x81 = %.3f'];   % \xCF\x81 = Greek rho in TeX text
txt = sprintf(fmt, alpha, beta, MAPE, max_pct_err, rho);

xpos = min(A) + 0.05*(max(A)-min(A));
ypos = max(B) - 0.05*(max(B)-min(B));
text(xpos, ypos, txt, ...
   'VerticalAlignment','top', ...
   'FontName','monospace', ...
   'FontSize', 35, ...
```

```
    'FontWeight','bold', ...
    'BackgroundColor','w', ...
    'Margin',6, ...
    'Interpreter','tex');   % ensures Greek rho renders

% (MATLAB) global-ish defaults for this session:
set(groot,'DefaultAxesFontSize',24);
set(groot,'DefaultTextFontSize',24);

% (Octave-compatible) use the root 0:
set(0,'DefaultAxesFontSize',24);
set(0,'DefaultTextFontSize',24);

% Per-object sizing:
xlabel('BenchSEE(A)','FontSize',50,'FontWeight','bold');
ylabel('SERT (B)','FontSize',50,'FontWeight','bold');
title(sprintf('Linear Fit (R^2 = %.4f)', R2), 'FontSize', 50, 'FontWeight','bold');
legend('Location','southeast','FontSize',50);
set(gca,'FontSize',50,'LineWidth',1.25);
set(gcf,'Color','w','Units','normalized','Position',[0.1 0.1 0.6 0.6]);

% ==================== helper functions ====================
function r = spearman_rho(x,y)
  x = x(:); y = y(:);
  ok = ~(isnan(x) | isnan(y));
  x = x(ok); y = y(ok);
rx = avg_tied_ranks(x);
ry = avg_tied_ranks(y);
rx = rx - mean(rx); ry = ry - mean(ry);
  r = (rx' * ry) / sqrt((rx' * rx) * (ry' * ry));
end

function r = avg_tied_ranks(v)
  [sv, idx] = sort(v);
  n = numel(v); r = zeros(n,1);
  i = 1;
  while i <= n
    j = i;
    while j <= n &&sv(j) == sv(i), j = j + 1; end
rank_ij = (i + j - 1) / 2;           % average rank for ties
    r(idx(i:j-1)) = rank_ij;
    i = j;
  end
end
```

3) Code for Fig. 3

```
% Data (A: BenchSEEindependent, B: SERT dependent)
A = [178.34; 182.44; 210.96; 215.81; 47.73; 53.5; 75.7];
B = [118.96; 139.1; 152.1; 155.02; 22.63; 25.46; 39.58];

% Design matrix (intercept + slope)
X = [ones(size(A)), A];

% Least squares (numerically stable)
theta = X \ B;                % same as (X'*X)\(X'*B) but better conditioned
beta  = theta(1);             % intercept
alpha = theta(2);             % slope

% Predictions and residuals
B_hat = X * theta;
res   = B - B_hat;

% Per-point percent errors and max |% error|
pct_err = abs((B - B_hat)./B) * 100;     % [%]
max_pct_err = max(pct_err);

% Fit quality
SS_tot = sum((B - mean(B)).^2);
SS_res = sum(res.^2);
R2     = 1 - SS_res/SS_tot;

% Error metrics
MAPE = mean(abs(res)./abs(B))*100; % mean absolute percentage error

% ---------- Spearman rank correlation (no toolboxes needed) ----------
% Try built-in corr if available; otherwise use fallback function.
if exist('corr','file')
  try
    rho = corr(A, B, 'type', 'Spearman');
  catch
    rho = spearman_rho(A, B);
  end
else
  rho = spearman_rho(A, B);
end
% ---------------------------------------------------------------------

% Display in console
fprintf('alpha (slope)   = %.4f\n', alpha);
fprintf('beta (intercept)= %+.4f\n', beta);
```

```
fprintf('R^2             = %.4f\n', R2);
fprintf('MAPE            = %.2f %%\n', MAPE);
fprintf('Max |%% error|  = %.2f %%\n', max_pct_err);
fprintf('Spearman rho    = %.4f\n', rho);

% Plot data + regression line + annotation
figure; hold on; grid on;
plot(A, B, 'bo', 'MarkerSize', 30, 'DisplayName', 'Data');

A_line = linspace(min(A), max(A), 200);
B_line = beta + alpha*A_line;
plot(A_line, B_line, 'r-', 'LineWidth', 2, 'DisplayName', 'B = \alpha A + \beta');

legend('Location','southeast');
xlabel('A'); ylabel('B');
title(sprintf('Linear Fit (R^2 = %.4f)', R2));

% Put equation, errors & Spearman on the plot
fmt = ['B = %.4f·A %+.4f\n' ...
       'MAPE = %.2f%%\n' ...
       'Max |%% err| = %.2f%%\n' ...
       'Spearman \xCF\x81 = %.3f'];   % \xCF\x81 = Greek rho in TeX text
txt = sprintf(fmt, alpha, beta, MAPE, max_pct_err, rho);

xpos = min(A) + 0.05*(max(A)-min(A));
ypos = max(B) - 0.05*(max(B)-min(B));
text(xpos, ypos, txt, ...
   'VerticalAlignment','top', ...
   'FontName','monospace', ...
   'FontSize', 35, ...
   'FontWeight','bold', ...
   'BackgroundColor','w', ...
   'Margin',6, ...
   'Interpreter','tex');   % ensures Greek rho renders

% (MATLAB) global-ish defaults for this session:
set(groot,'DefaultAxesFontSize',24);
set(groot,'DefaultTextFontSize',24);

% (Octave-compatible) use the root 0:
set(0,'DefaultAxesFontSize',24);
set(0,'DefaultTextFontSize',24);

% Per-object sizing:
xlabel('BenchSEE(A)','FontSize',50,'FontWeight','bold');
ylabel('SERT (B)','FontSize',50,'FontWeight','bold');
title(sprintf('Linear Fit (R^2 = %.4f)', R2), 'FontSize', 50, 'FontWeight','bold');
legend('Location','southeast','FontSize',50);
set(gca,'FontSize',50,'LineWidth',1.25);
set(gcf,'Color','w','Units','normalized','Position',[0.1 0.1 0.6 0.6]);

% ==================== helper functions ====================
function r = spearman_rho(x,y)
  x = x(:); y = y(:);
  ok = ~(isnan(x) | isnan(y));
  x = x(ok); y = y(ok);
rx = avg_tied_ranks(x);
ry = avg_tied_ranks(y);
rx = rx - mean(rx); ry = ry - mean(ry);
  r = (rx' * ry) / sqrt((rx' * rx) * (ry' * ry));
end

function r = avg_tied_ranks(v)
  [sv, idx] = sort(v);
  n = numel(v); r = zeros(n,1);
  i = 1;
  while i <= n
    j = i;
    while j <= n &&sv(j) == sv(i), j = j + 1; end
rank_ij = (i + j - 1) / 2;           % average rank for ties
    r(idx(i:j-1)) = rank_ij;
    i = j;
  end
end
```

III. Results

As shown in Fig. 1, Fig. 2, and Fig. 3,BenchSEE and SERT seem interchangeable as the Spearman coefficient (ρ) is 1, and MAPE and Max |% error| are all less than 10%.

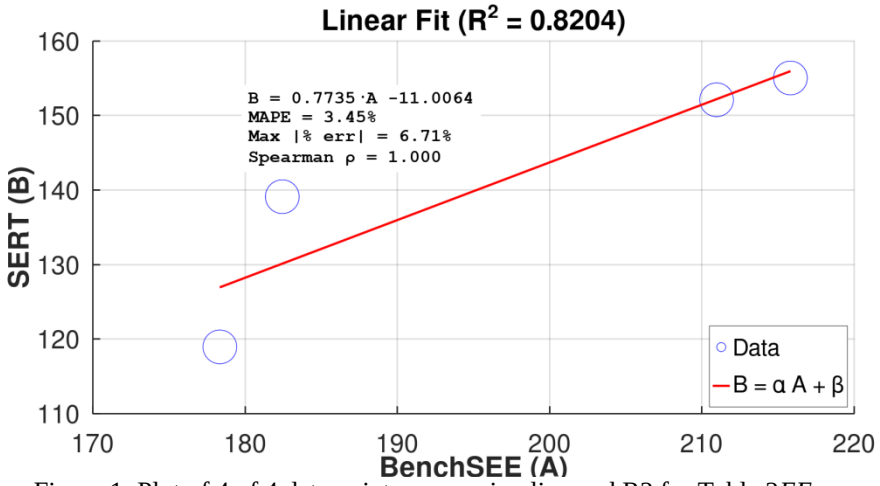

Figure 1: Plot of 4 of 4 data points, regression line and R2 for Table 2$EE_{tot.}$

Fig. 2 - with one less data pair than Fig. 1 - is generated to show the flexibility of the implementation.

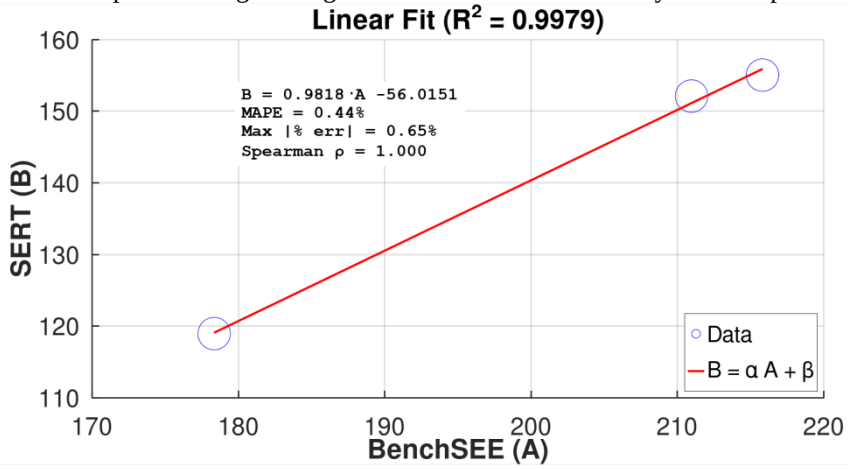

Figure 2: Plot of 3 of 4 the data points excluding SERT2 and BenchSEE2, regression line and R2 for Table 2$EE_{tot.}$

Fig.3: shows indications of adding other preliminary EE_{tot} results.

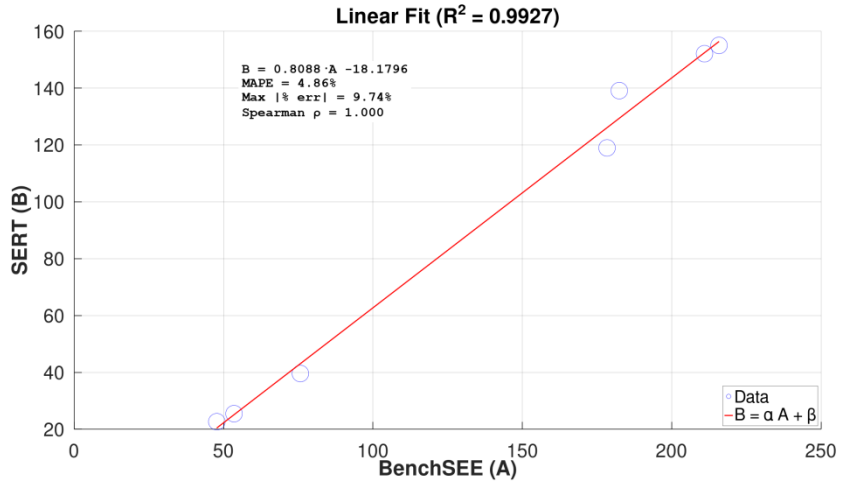

Figure 3: Plot including three additional data points to Table 2$EE_{tot.}$, regression line and R2.

IV. Discussion

The Spearman ranking coefficient is 1 for all investigations and this raises the possibility for ranking equivalence, although the limited dataset prevents robust conclusions as the plausible value of the coefficient could be much lower. Likely around 30 datapoints are required for a clear confidence in the ranking equivalence.

The mean absolute percentage error (MAPE) averages the miss telling how much the predictions differ (in %) compared to the true value. MAPE is unitless and scale-free making it comparable across datasets and variables. However, MAPE overweighs small differences but is adequate for overall accuracy and "how good on average". MAPE is advantageously combined with Max |% error| to get a complete picture.

The Max |% error| is defined as the worst-case relative miss in %. The indicator points out any outlier that violates the set tolerance. Max |% error| can be sensitive to outliers but is enough for acceptance choices and questions like "is the tolerance limit ever violated?" 3-5% gives strong assurance, 5-10% is acceptable for engineering comparisons, while 10-15% Max |% error| will have at least one data pair failing.

The present investigation offers a preliminary signal, but the low number of datapoints precludes firm conclusions. Anyway, the analysis code can be reused for any number of data pairs. The framework could be expanded with the so called P95 % error which answers "95% of cases have this absolute percent error".

V. Conclusion

For the first time, an implementation for test functional equivalence of server EE benchmarking tools is developed. The implementation is applied to limited tool comparison suggesting that BenchSEE and SERT are provisionally ranking and calibration equivalent. However, further research with larger samples is required to substantiate the conclusion. The implementation works well. However, even though the ranking and calibration equivalences seem apparent, more studies including more server types are necessary to clearly establish the functional equivalence between the EE benchmark tools.

References

[1]. A.S.G. Andrae, From an Environmental Viewpoint Large ICT Networks Infrastructure Equipment must not be Reused, *WSEAS Transactions on Environment and Development, 19,*2023, 375–382.DOI: https://doi.org/10.37394/232015.2023.19.34

[2]. N. Rteil, K. Burdett, S. Clement, A. Wynne, and R. Kenny. Balancing power and performance: A multi-generational analysis of enterprise server bios profiles. *In 2022 international conference on green energy, computing and sustainable technology (gecost),* 2022, 81–85. IEEE. DOI: 10.1109/GECOST55694.2022.10010599.

[3]. M. Meissner, K.D. Lange, J. Arnold et al.SPEC Efficiency Benchmark Development: How to Contribute to the Future of Energy Conservation. *In Companion of the 2022 ACM/SPEC International Conference on Performance Engineering* 2022, 21–24. DOI: https://doi.org/10.1145/3491204.3527492

[4]. K.W. Cameron. Adventures Beyond Amdahl's Law: How Power-Performance Measurement and Modeling at Scale Drive Server and Supercomputer Design. *Journal of Computer Science and Technology, 38(1),* 2023, 80–86. DOI: https://doi.org/10.1007/s11390-022-2950-7

[5]. W. Lin, X. Luo, C. Li, J. Liang, G. Wu, and K. Li,An energy-efficient tuning method for cloud servers combining DVFS and parameter optimization,*IEEE Transactions on Cloud Computing, Vol. 11(4),*2023, 3643–3655.DOI: 10.1109/TCC.2023.3308927

[6]. L. Guo, Y. Wang, Y. Zhang, C. Zhou, K. Xu, and S. Wang, A new model and testing verification for evaluating the carbon efficiency of server, *KSII Transactions on Internet & Information Systems, 17(10),* 2023, 2682–2700.DOI: https://doi.org/10.3837/tiis.2023.10.005

[7]. J. Liang, W. Lin, Y. Xu, Y. Liu, R. Mo, and X. Luo. Energy-aware parameter tuning for mixed workloads in cloud server, *Cluster Computing, Vol. 27(4),*2024, 4805–4821.DOI: https://doi.org/10.1007/s10586-023-04212-6

[8]. W. Lin, W. Lin, J. Lin, H. Zhong, J. Wang, and L. He. A multi-agent reinforcement learning-based method for server energy efficiency optimization combining DVFS and dynamic fan control, *Sustainable Computing: Informatics and Systems, 42,* 2024, 100977.DOI: https://doi.org/10.1016/j.suscom.2024.100977

[9]. G. Wu, H. Wang, W. Lin, R. Mo, and X. Luo. FS-DBoost: cross-server energy efficiency and performance prediction in cloud based on transfer regression, *Cluster Computing, 27(6),* 2024, 7705–7719. DOI: https://doi.org/10.1007/s10586-024-04370-1

[10]. W. Lin, J. Lin, J. Li, M. Guo, L. He, and W. Lin. An adversarial reinforcement learning method to enhance server energy efficiency via DVFS and dynamic fan speed management. *Computing, 107(160),*2025, 1–30.DOI: https://doi.org/10.1007/s00607-025-01497-w

[11]. M. Mo, W. Lin, H. Zhong, M. Xu, and K. Li. A Cross-Workload Power Prediction Method Based on Transfer Gaussian Process Regression in Cloud Data Centers, *IEEE Transactions on Cloud Computing, 13(3),*2025, 910–921.DOI: 10.1109/TCC.2025.3575790

[12]. V. Pandey, P. Saini, M. Gupta, and A. Bhardwaj. Dynamic Job Scheduling for Energy Savings in YARN: A Comparative Study With Benchmark Workloads, *Concurrency and Computation: Practice and Experience, 37(15–17),*2025, e70144.DOI: https://doi.org/10.1002/cpe.70144

[13]. J. von Kistowski, M. Meißner, K.D. Lange and J.A. Arnold. Server Energy Efficiency Benchmarks. *In Systems Benchmarking: For Scientists and Engineers*, 2025, 247–265. Cham: Springer Nature Switzerland. DOI: https://doi.org/10.1007/978-3-031-85634-1_11

[14]. European Telecommunications Standards Institute. Energy Efficiency measurement methodology and metrics for heterogeneous servers. 2025. https://www.etsi.org/deliver/etsi_es/204000_204099/204083/01.01.01_60/es_204083v010101p.pdf. Accessed 5November 2025.

[15]. J.J. Workman. A review of calibration transfer practices and instrument differences in spectroscopy. *Applied spectroscopy, Vol. 72(3),* 2017, 340–365.DOI: https://doi.org/10.1177/0003702817736064

[16]. J.C. Chow, X. Wang, B.J. Sumlin, S.B.Gronstal, L.W.A. Chen, D.L. Trimble, S.D Kohl, S.R. Mayorga, G. Riggio, P.R. Hurbain, M. Johnson, R. Zimmermann, and J.G. Watson, Optical calibration and equivalence of a multiwavelength thermal/optical carbon analyzer. *Aerosol and Air Quality Research, Vol. 15(4),* 2015, 1145–1159. DOI: https://doi.org/10.4209/aaqr.2015.02.0106

[17]. H. Carstens, X. Xia, andS. Yadavalli. Low-cost energy meter calibration method for measurement and verification. *Applied energy, 188,* 2017, 563–575. DOI: https://doi.org/10.1016/j.apenergy.2016.12.028

[18]. A. Cultrera, G. Germito, D. Serazio, F. Galliana, B. Trinchera et al. Active Energy Meters Tested in Realistic Non-Sinusoidal Conditions Recorded on the Field and Reproduced in Laboratory. *Energies, 17(6),* 2024, 1403. DOI: https://doi.org/10.3390/en17061403

[19]. A. Pajankar and S. Chandu. GNU Octave by Example. DOI: https://doi.org/10.1007/978-1-4842-6086-9